Universidad de Alcalá

Escuela Politécnica Superior

Grado en Ingeniería en Sistemas de Telecomunicación

Trabajo Fin de Grado

Desarrollo de un simulador de Radios sobre IP para pruebas de sistemas de comunicación por voz

Autor: Javier León Íñigo

Tutor: Javier Macías Guarasa

Cotutora: Yajaira Salgado Lorenzo

2019

UNIVERSIDAD DE ALCALÁ
ESCUELA POLITÉCNICA SUPERIOR

GRADO EN INGENIERÍA EN SISTEMAS DE TELECOMUNICACIÓN

Trabajo Fin de Grado

Desarrollo de un simulador de Radios sobre IP para pruebas de sistemas de comunicación por voz

Autor: Javier León Íñigo

Tutor: Javier Macías Guarasa

Cotutora: Yajaira Salgado Lorenzo

TRIBUNAL:

Presidente: José Manuel Villadangos Carrizo

Vocal 1º: Juan Jesús García Domínguez

Vocal 2º: Javier Macías Guarasa

CALIFICACIÓN:

FECHA:

A mi familia, por ayudarme siempre a seguir intentándolo…

"El fracaso es una gran oportunidad para empezar otra vez con más inteligencia."

Henry Ford

Agradecimientos

En primer lugar, agradecer a Javier Macías el tiempo dedicado en mi proyecto. Con su apretada agenda, ha conseguido sacar tiempo para ayudarme con mis dudas y problemas que han ido surgiendo durante el desarrollo de este proyecto. Agradecer también a Yajaira Salgado, mi responsable en Indra S.A., su ayuda para realizar este proyecto como sus consejos a la hora de realizar esta memoria.

También quiero agradecer la inestimable ayuda de todos los amigos y compañeros que he hecho durante esta maravillosa etapa de mi vida que ha sido la universidad. En concreto, darle las gracias a Fang por ser mi "pepito grillo", Sofía, Aarón, Borja y Celia por hacer más amenas las horas y horas en la biblioteca.

Me gustaría también agradecer toda su paciencia infinita a Fabianna, que es la persona que más me ha tenido que aguantar en los momentos de mayor estrés y siempre ha sabido sacarme una sonrisa cuando más lo necesitaba. A mis amigos de toda la vida Alba, Eduardo, Miguel, Javi, Alejandro, Gerardo, Carla y Carmen por todos los buenos consejos y motivación extra recibidos durante estos años.

Y por último y no por ello menos importante, a mis padres Oscar e Isabel por todo el apoyo a lo largo de toda la carrera y sus continuas e importantes lecciones sobre la vida. Si estoy donde estoy en la actualidad se lo debo a su continuo esfuerzo y dedicación en mi educación y desarrollo personal. A mi hermano Ignacio por ser una continua fuente de motivación y un ejemplo de continua superación personal y profesionalmente hablando.

A todos ellos y otra mucha gente que no se menciona aquí, solo puedo deciros: GRACIAS.

Índice General

Índice de Figuras

Resumen

En este Trabajo Fin de Grado se propone la implementación de una herramienta capaz de simular radios sobre IP para entornos de desarrollo y pruebas en sistemas de comunicación por voz en entornos críticos, como puede ser las comunicaciones en el ámbito de defensa, ATM o seguridad.

Para el progreso de esta herramienta, se utiliza el sistema de comunicaciones de voz Garex300, desarrollado en su integridad por Indra Sistemas S.A. De este entorno se van a obtener todos los datos necesarios para la simulación de las radios sobre IP

Palabras clave:

RIS (*Radio IP Simulator*): Nombre que recibe la herramienta que se procede a desarrollar en este proyecto.

ATM (*Air Traffic Management*): Estas siglas, en aviación, abarcan todos los sistemas que ayudan a las aeronaves desde que despegan del aeropuerto, transitan por el aire y aterrizan en el aeropuerto destino.

VoIP (*Voice over IP*): Es el conjunto de recursos que hacen posible que la señal de voz viaje a través de Internet empleando protocolo IP.

SIP (*Session Initiation Protocol*): Protocolo de señalización para conferencias, telefonía, presencias, notificación de eventos y mensajería instantánea a través de Internet. Este protocolo considera a cada conexión como un par y se encarga de negociar las capacidades entre ellos, con una sintaxis simple, similar a HTTP, y un sistema de autentificación de pregunta/respuesta.

RTP (*Real Time Protocol*): Protocolo utilizado para la transmisión de información en tiempo real, como el video y/o el audio. Es un protocolo que trabaja en UDP, menos seguro que TCP, pero da mejores prestaciones en cuanto a latencia se refiere.

Abstract

In this Final Degree Project, we propose the implementation of a tool capable of simulating radio over IP for development and test environments in voice communication systems in critical environments, such as communications in the field of defense, ATM or security.

For the progress of this tool, the Garex300 voice communications system, fully developed by Indra Sistemas S.A. is used. From this environment all the data necessary for the simulation of the radios over IP will be obtained.

Keywords:

RIS (Radio IP Simulator): This is the name given to the tool developed in this project.

ATM (Air Traffic Management): These acronyms, in aviation, cover all the systems that help aircraft since they take off from the airport, transit through the air and land at the destination airport.

VoIP (Voice over IP): It is the set of resources that make it possible for the voice signal to travel through the Internet using IP protocol.

SIP (Session Initiation Protocol): Signaling protocol for conferences, telephony, presences, notification of events and instant messaging over the Internet. This protocol considers each connection as a pair and is responsible for negotiating the capabilities between them, with a simple syntax, like HTTP, and a question / answer authentication system.

RTP (Real Time Protocol): Protocol used for the transmission of information in real time, such as video and audio. It is a protocol that works in UDP, less secure than TCP, but gives better performance in terms of latency.

Resumen Extendido

Las comunicaciones vía radio convencional operando en bandas como VHF (*Very High Frequency*) o UHF (*Ultra High Frequency*) son muy utilizadas, aún hoy en día, por empresas, radioaficionados y organismos de seguridad público y privado, jugando un papel importante en acciones de vigilancia, protección y control, sobre todo donde la infraestructura de telefonía móvil e Internet no proporciona la conectividad o calidad necesaria.

Las comunicaciones bidireccionales de radio analógica, moduladas en FM y operando en las bandas VHF y UHF, son conocidas como "comunicaciones simples" por carecer de "inteligencia" de control. Esta falta de control provoca que las comunicaciones realizadas sean en la modalidad half duplex, lo que conlleva que, por ejemplo, en una comunicación entre dos operadores radio, mientras uno transmite el otro operador se ve forzado a ser el receptor hasta que el operador transmisor libere la línea. Otro problema añadido a esta falta de control es la fácil intercepción y el descifrado de posibles mensajes con información sensible, ya que solo nos haría falta un receptor radio con el demodulador correcto en la frecuencia de transmisión.

A raíz de estos problemas, surge la idea de RoIP (*Radio over IP*, Radio sobre IP). Esta tecnología mezcla los sistemas radio convencionales con VoIP (*Voice over IP*, Voz sobre IP), permitiendo un nuevo mapa de posibilidades de comunicación a través de radio. Podemos diseñar un mapa básico como dos radios que se comunican a través de un radio enlace en el que la mensajería es transmitida en formato IP con los distintos protocolos de comunicación que requiere el sistema (SIP, RTP, RTPD) o un sistema más complejo en el que, a través de una radio, podamos comunicarnos a un teléfono IP gracias a centralitas y *Gateway* que permiten la integración de ambos mundos, como se puede observar en la Figura 1.

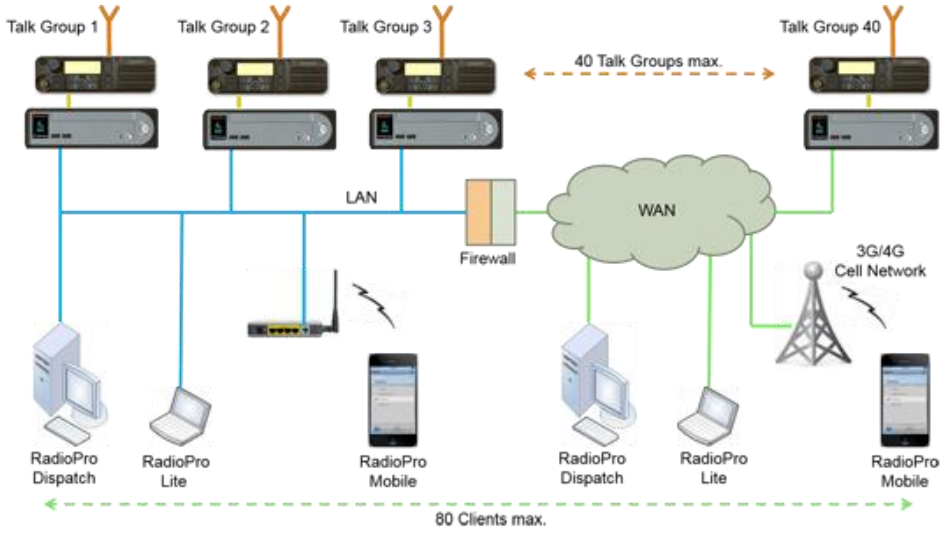

Figura 1: Ejemplo de sistema RoIP

En este nuevo escenario, se abre la puerta a mejorar las comunicaciones radio de misión crítica, como radio militar, tráfico aéreo y marítimo, seguridad y resto de organismos en el que la comunicación sea esencial tanto la calidad como la seguridad de esta. Además de contar con protocolos de seguridad para el mensaje, gracias a la tecnología RoIP podemos aumentar considerablemente el radio de alcance de una comunicación radio, gracias a las centralitas y *Gateway* anteriormente comentados con los que podemos diseñar amplias redes de comunicación.

Para el desarrollo de este nuevo escenario que se nos presenta, y teniendo en cuenta que hablamos de entornos críticos, necesitamos sistemas que sean completamente seguros y fiables. Conseguir que estos sistemas cumplan estos requisitos, requiere de un completo trabajo de desarrollo y pruebas donde se tenga en cuenta cada escenario posible que el sistema pueda padecer. Es en esta fase de desarrollo donde se quiere realizar una herramienta, para facilitar o mejorar las posibles pruebas que el sistema tenga que soportar.

Esta herramienta, que denominaremos RIS (*Radio IP Simulator*), contará con un interfaz de usuario para modificar una gran cantidad de variables y/o funcionalidades que la radio pueda tener, como son: BSS (*Best Sound Selection*), añadir retardos, introducción de audio desde el RIS contra el SCV (Sistema de Comunicación por Voz), deshabilitar la sesión SIP (*Session Initiation Protocol*) por el motivo que el usuario desee y otras tantas funcionalidades.

Para que el RIS funcione correctamente, debe estar conectado a un SCV constantemente. Gracias a esto, se le proporcionan al RIS las cualidades que debe poseer cada radio a simular y, a su vez, proporciona al SCV las sesiones SIP de las radios simuladas para poder manipularlas.

El SCV con el que se va a trabajar para el desarrollo de esta herramienta será el Garex300 [2], desarrollado por Indra S.A. Como se muestra en la Figura 2, esta es la arquitectura que presenta este SCV en concreto.

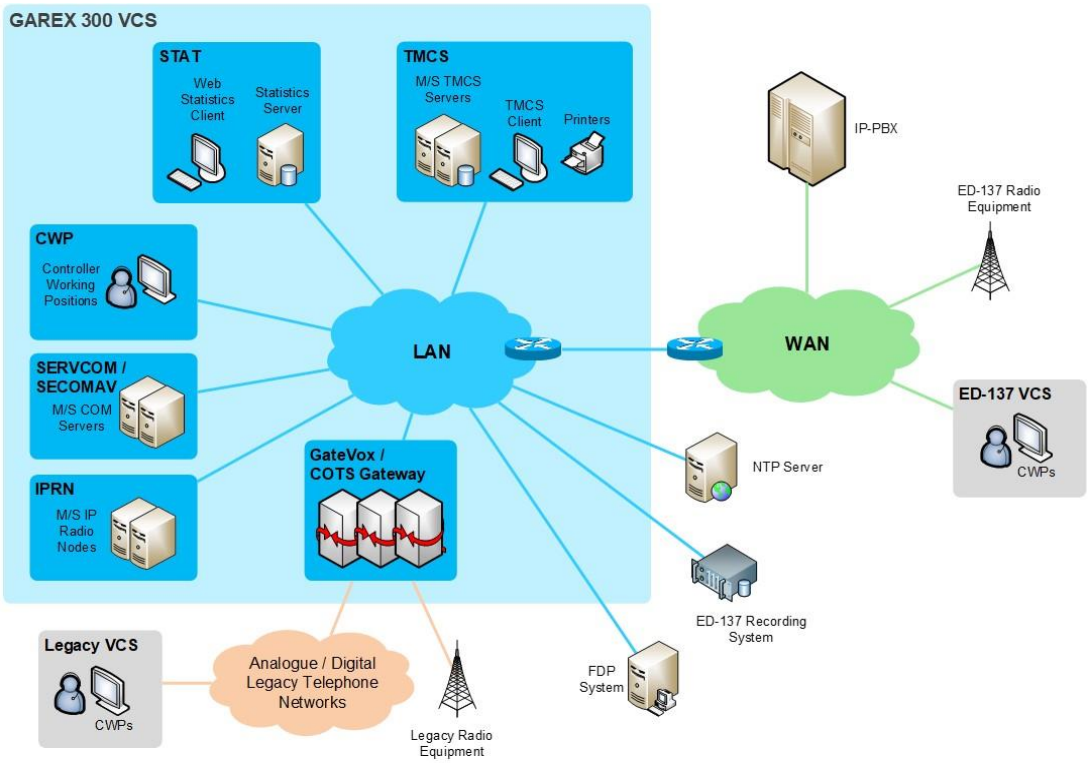

Figura 2: Arquitectura de Garex300

Acrónimos

- RIS (*Radio IP Simulator*)
- VoIP (*Voice over IP*)
- RoIP (Radio over IP)
- VCS/SCV (*Voice Communication System*, Sistema de Comunicación por Voz)
- SIP (*Session Initiation Protocol*)
- RTP (*Real Time Protocol*)
- ATS (*Air Traffic Services*)
- ITU (*International Telecommunication Union*)
- PTT (*Push To Talk*, Pulsar Para Hablar)
- EUROCAE (*EURopean Organisation for Civil Aviation Equipment*)
- TMCS (*Technical Monitoring and Configuration Server*)
- IPRN (*IP Radio Node*)
- ER (Equipo Radio)
- UR (Unión Radio)
- Tx (Transmisor)
- Rx (Receptor)
- TxRx (Transceptor)
- FC (Frecuencia de control)
- CWP (*Controller Working Position*)
- SERCOMAV (SERvicio de COMunicación AVanzado)
- CF (*Conference Focus*)
- SQ (*Squelch*)
- HMI (*Human Machine Interface*)
- SDP (*Session Description Protocol*)
- IDE (*Integrated Development Environment)*
- GUI (*Graphical User Interface)*
- HW (*HardWare*)
- SW (*SoftWare*)
- MW (*MiddleWare*)

1 Motivación y objetivos

1.1 Introducción

El objetivo fundamental de este proyecto es el diseño, desarrollo y operativa de una herramienta de simulación de radios sobre IP. Esta herramienta estará enfocada a sistemas de comunicación por voz, como pueden ser los SCV usados por los controladores aéreos.

1.2 Objetivos

Dicha herramienta, está focalizada en la etapa de desarrollo y testeo de calidad de dichos sistemas de comunicación por voz para entornos críticos, ya que se podría utilizar para comprobar la robustez del sistema, el correcto funcionamiento de distintas funcionalidades que el sistema de comunicación pueda poseer y probar los distintos estándares que la ITU (*International Telecommunication Union*) y EUROCAE [1] definen para esta tecnología.

El RIS se desarrollará con una interfaz para facilitar la manipulación que quiera darle el usuario. También necesitará la conexión directa con el sistema de comunicación por voz para obtener de este los datos de las radios que se quieren simular y sus funcionalidades.

Durante este desarrollo, se partirá de archivos de configuración del sistema de comunicación por voz Garex300, desarrollado por Indra Sistemas S.A.

1.3 Estructura

La estructura de la memoria es la que se detalla a continuación:

- Capítulo 1: Motivación y objetivos.
- Capítulo 2: Diseño del sistema.
- Capítulo 3: Desarrollo de la herramienta RIS.
- Capítulo 4: Visualización y funcionamiento.
- Capítulo 5: Conclusiones y líneas futuras.

2 Diseño del sistema

2.1 Introducción

En esta sección previa al desarrollo del proyecto, se realizará un estudio para conocer la estructura general del sistema, explicando brevemente la funcionalidad de cada una de las partes que van a formar parte del simulador de radio sobre IP. La arquitectura de dicha herramienta se puede observar en la Figura 3.

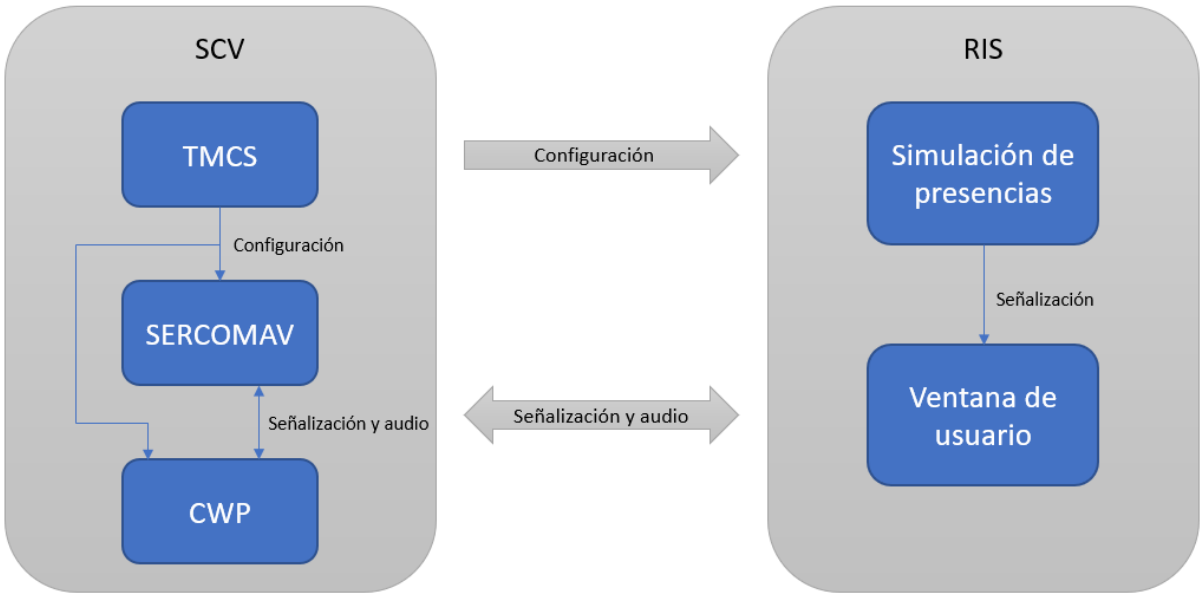

Figura 3: Arquitectura del sistema

2.2 Sistema de Comunicación por Voz (SCV)

2.2.1 TMCS

En primer lugar, se creará un archivo de configuración en TMCS (*Technical Monitoring and Configuration Server*), que forma parte del SCV Garex300. Este programa permite al usuario diseñar y configurar la totalidad de las características del centro de comunicaciones que va a gestionar, desde las IPs que el sistema va a utilizar hasta los abonados ATS (*Air Traffic Services*) que dicho centro de comunicaciones va a asignar, pasado por las configuraciones de los equipos radio (ER). Para configurar estos ER, se requiere de los siguientes campos:

- Nombre que se va a asignar al ER.
- Asignar la topología del ER, pudiendo ser transmisor (Tx), receptor (Rx) o transceptor (TxRx).
- SIP-URI [3] de dicho ER. Esta SIP-URI es el esquema de direccionamiento SIP que se requiere para comunicarse con un agente externo vía SIP. La nomenclatura para este campo debe tener el siguiente formato: nombreSIPdelER@IPdelER, como, por ejemplo, Tx1@192.168.1.217.
- Hay que indicar que SERCOMAV, el cual explicaremos más adelante, va a realizar las gestiones de las diferentes sesiones de este ER.

Una vez establecida la información requerida para configurar los ER, se nos permite la opción de agrupar los ER en Uniones Radio (UR). Agrupando los ER en UR podemos configurarlos para que trabajen conjuntamente Tx con Rx o, incluso, configurar en modo Main/Stand-by (M/S). Obligatoriamente, en cada UR debe haber como mínimo un Rx o un TxRx, lo que nos permite conseguir las siguientes topologías de UR: Rx, TxRx, Tx+Rx, Rx – Rx, TxRx – TxRx y Tx+Rx – Tx+Rx.

Teniendo establecido ya las UR, se procede a asignar dichas UR en Frecuencias de Control (FC). Con las FC se nos permite agrupar todas las UR que deseemos bajo una misma frecuencia, donde podemos asignarle cualquier rango disponible entre 000.000 y 999.999 MHz en la configuración, aunque EUROCAE delimita el rango de frecuencias de trabajo entre 117.975 y 136.975 MHz para las comunicaciones con aeronaves.

Con las FC ya configuradas, ahora debemos asignar dichas FC a las distintas posiciones de trabajo del controlador (CWP, *Controller Working Positions*). Asignando las FC a las distintas CWPs, elemento que explicaremos más adelante, el controlador obtiene la manera de comunicarse vía radio IP con las distintas aeronaves que tenga que gestionar.

Con todos los parámetros necesarios y deseados para la configuración, el TMCS genera ficheros de configuración que transmite al resto de componentes del SCV, entre los que se encuentra el SERCOMAV, CWP y el RIS.

2.2.2 SERCOMAV

Una vez tenemos la configuración de las radios que deseamos realizada e implementada en el sistema, entra en funcionamiento tanto el SERCOMAV como la CWP. A continuación, explicaremos que es el SERCOMAV y su funcionalidad.

El SERCOMAV consta de tres servicios distintos, que son los siguientes:

I. Proxy. El SERCOMAV realiza la función de un proxy, el cual hace de sistema intermedio entre todos los elementos del Garex300, ya sea entre CWPs, CWP contra abonados externos del sistema, etc. Esto facilita el rendimiento, registro del tráfico total y control de acceso del sistema. Solo se utiliza para el enrutado de la mensajería y las comunicaciones de inicio de sesión SIP, ya que la mensajería de audio es directa entre destinos.

II. IPRN. Este elemento presta el servicio de gestión de las sesiones SIP establecidas con las radios IP. Este servicio provoca que podamos aumentar considerablemente el número de sesiones SIP establecidas a las radios, las cuales están limitadas por un valor oscilante entre 3 y 10 sesiones, dependiendo de cada fabricante.

III. CF (*Conference Focus*). Con este servicio, el SERCOMAV es capaz de gestionar las conferencias que se puedan establecer en el sistema. Pueden ser conferencias entre abonados internos o incluir abonados externos.

Para la herramienta que vamos a desarrollar, vamos a necesitar dos de los tres servicios que el SERCOMAV nos presta, concretamente el Proxy y el IPRN. El Proxy es necesario a la hora de obtener los archivos de configuración realizados en el TMCS, ya que es el Proxy el que se encarga de distribuir esos archivos desde el TMCS al resto de elementos del sistema. Con esta información, el RIS sabrá cuantas radios IP debe simular y las características de estas. El IPRN también es necesario para establecer las sesiones SIP con los ER simulados por el RIS. Para que el sistema detecte correctamente estas radios, el RIS debe ser capaz de establecer sesiones SIP con el IPRN en los diferentes modos posibles.

2.2.3 CWP

La CWP representa el HMI (*Human Machine Interface*) con el que el controlador aéreo puede operar con el sistema realizando tanto comunicaciones de telefonía como de radio y utilizar las diferentes funcionalidades de las que dispone. Este HMI se divide básicamente en dos partes: telefonía y radio. La parte que nos ocupa en esta memoria es la correspondiente a la parte radio.

En esta parte, podemos encontrar las diferentes teclas que representan las FC (Frecuencias de Control) anteriormente configuradas en el TMCS. Cada tecla indica el tipo de selección radio (Rx o Tx-Rx) y, mediante una configuración de colores, muestra el estado de la radio (reposo, recepción de audio, fallo parcial, etc.).

Cada CWP dispone de varias páginas radio disponibles, pero solo una se muestra en pantalla. Las FC que se muestren en dicha pantalla serán las que establezcan sesión SIP con el SERCOMAV, por lo que, si una página tiene cinco FC, la CWP establecerá cinco sesiones SIP con el SERCOMAV, una por cada tecla de FC.

2.3 Protocolos de comunicación

En este apartado se procede a explicar los protocolos de comunicación entre el SCV y el RIS para comprender en su totalidad el funcionamiento que debe adquirir el RIS y cómo comportarse. En la siguiente figura podemos ver los distintos protocolos que tienen relevancia en la tecnología VoIP.

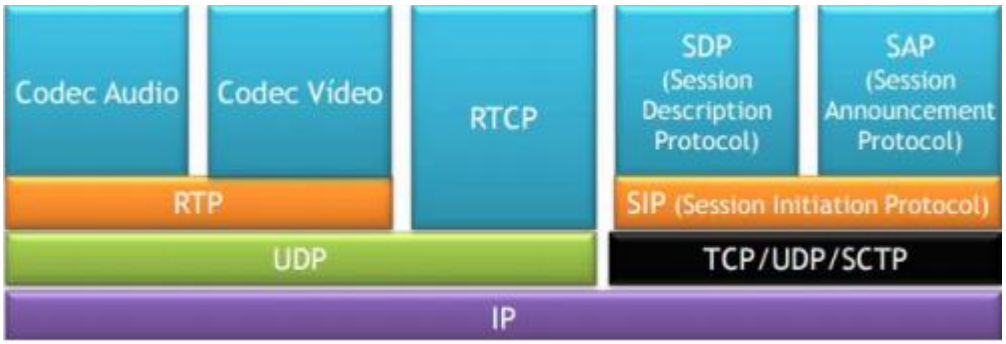

Figura 4: Torre de protocolos de VoIP

2.3.1 Session Initiation Protocol (SIP)

El protocolo SIP fue diseñado con la intención de ser el estándar para la iniciación, modificación y finalización de sesiones interactivas de usuario [4]. Al ser un protocolo de comunicación segura del tipo petición – respuesta, la capa de transporte que usa es la TCP además de un puerto de comunicación determinado, el cual suele ser el puerto 5060. Dentro de sus múltiples funciones posibles, vamos a focalizarnos en el uso que se le da en este entorno y explicar su operativa en el mismo.

La mensajería SIP dispone de varios métodos de petición. Entre todos ellos, destacamos los siguientes:

- *INVITE*. Mensaje de iniciación de establecimiento de comunicación o modificación de esta.
- *ACK*. Mensaje para confirmar una respuesta del colateral.
- *OPTIONS*. Mensaje utilizado por un cliente SIP para consultar a otro sobre sus capacidades y descubrir los métodos de comunicación soportados.
- *BYE*. Mensaje para finalizar una sesión previamente establecida.
- *CANCEL*. Mensaje utilizado para cancelar una petición.
- *REGISTER*. Mensaje usado para registrar o eliminar un usuario SIP de un servidor de registro.
- *SUBSCRIBE*. Mensaje para la obtención de estadísticas.

Tras cada mensaje de petición, el protocolo SIP envía un mensaje de estado o respuesta. Estos mensajes están en formato código, y se pueden clasificar de la siguiente manera:

- 1XX. Usado para indicar un estado temporal, como "100 *Trying*" o "180 *Ringin*".
- 2XX. Utilizado para indicar un estado final exitoso, como "200 *OK*".
- 3XX. Utilizado para indicar redireccionamiento. Contiene la información sobre la nueva ubicación del usuario o servicios alternativos.
- 4XX. Usado para indicar un error debido al cliente, ya sea en la petición u otro proceso generado por el usuario SIP.
- 5XX. Usado para indicar un error debido al servidor.
- 6XX. Usado para indicar un fallo global del sistema, como puede ser una repentina desconexión.

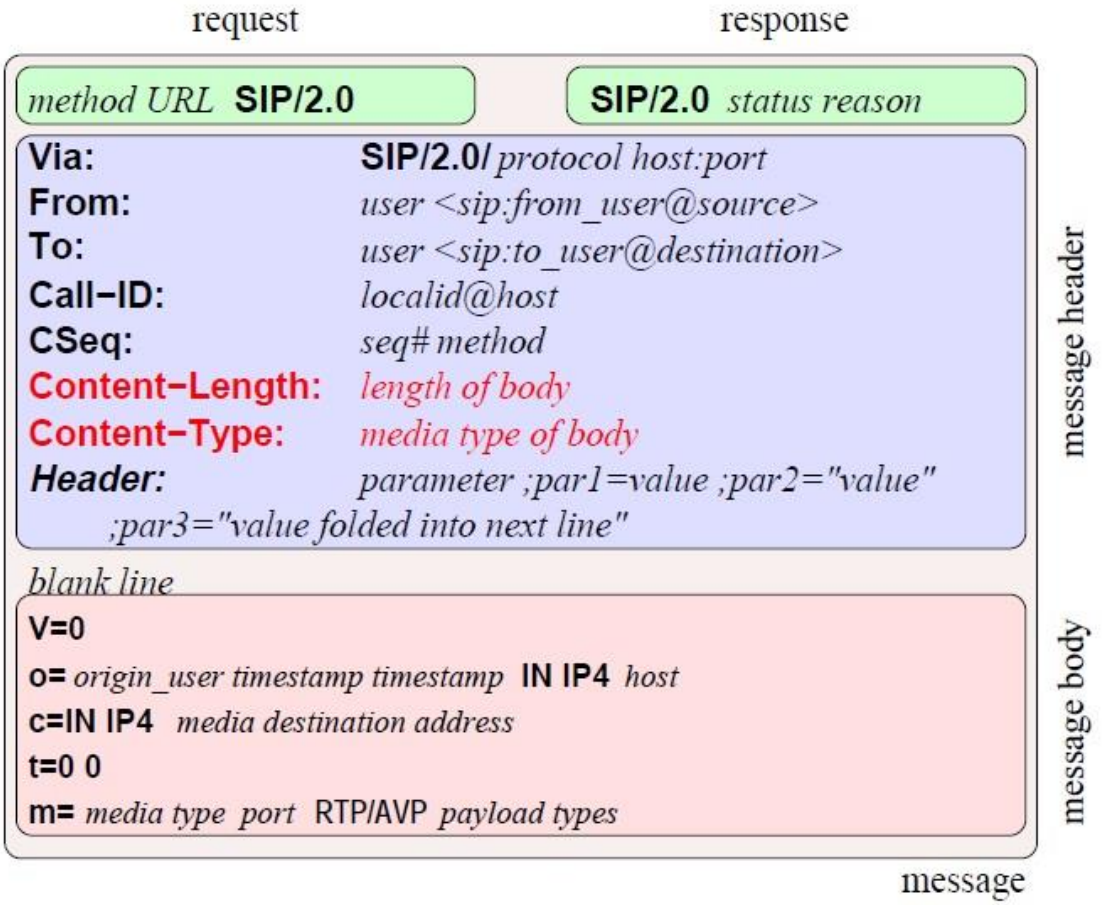

Figura 5: Ejemplo para mensajería SIP

Como podemos observar en la Figura 5, la mensajería del protocolo SIP tiene una cabecera (parte morada de la imagen) que es común tanto para peticiones como para respuestas, ya que solo varía el primer campo (color verde en la imagen). En esta cabecera podemos distinguir, obviando que el primer campo es el tipo de petición o respuesta, ocho campos. Estos campos son:

- *Via*: Indica el protocolo de transporte a usarse, la IP y puerto a dónde se deberán dirigir las respuestas.

- *From*: Indica la SIP URI del origen y puede ir acompañado del nombre del usuario.

- *To*: Indica la SIP URI del destinatario y puede ir acompañado del nombre del usuario.

- *Call-ID*: Muestra el ID global del diálogo. Será el mismo para todas las peticiones y respuestas dentro de la misma sesión.

- *CSeq*: Consiste en un número secuencial y el nombre del método. Es utilizado para ordenar las transacciones dentro de un diálogo.

- *Content-Length*: Representa el tamaño en bytes del cuerpo del mensaje.

- *Content-Type*: Describe el cuerpo del mensaje.

También podríamos encontrar otros campos como:

- *Max-Forwards*: Usado para limitar el número máximo de *proxys* o *Gateway* que pueden reenviar la petición al siguiente servidor. El número disminuirá en uno por cada servidor que reenvíe la petición. Si alcanza el valor 0, no se reenvía.

- *Contact*: Muestra el SIP URI que puede ser usado para contactar a este usuario en siguientes peticiones.

- *User-Agent*: Indica el modelo, marca e información particular del fabricante del terminal.

- *Allow*: Lista de métodos soportados por el usuario que ha generado el mensaje.

Como podemos observar en la Figura 5, hay también un apartado de cuerpo del mensaje. En esta parte, entra en juego el protocolo SDP (*Session Description Protocol*), que se encarga de contener la información del medio utilizado, como puede ser la versión del protocolo, códecs disponibles para la comunicación, ancho de banda a utilizar, ajustes de zona horaria, claves de cifrado y un largo etcétera de posibilidades para cada tipo de sesión SIP que deseemos establecer.

Las sesiones SIP que vamos a gestionar pueden diferenciarse en dos tramos: comunicaciones entre radio – SERCOMAV y las comunicaciones SERCOMAV – CWP. El primer vano se caracteriza por que el SERCOMAV realiza un INVITE a la radio en modo Idle (Reposo) automáticamente, para así confirmar que la radio está operativa. A continuación, se muestra un ejemplo de dicho establecimiento.

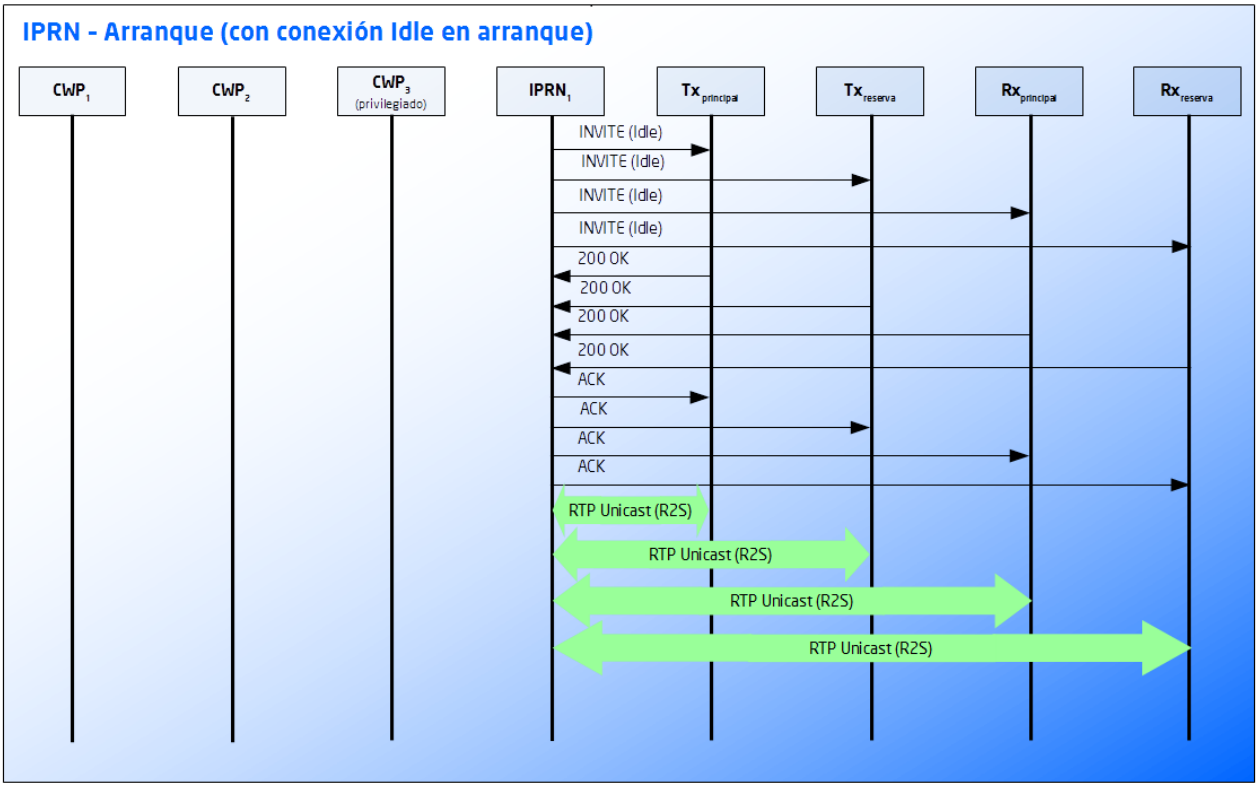

Figura 6: Establecimiento de sesiones SIP al iniciar el IPRN

Una vez se haya realizado el establecimiento de la sesión SIP, el controlador podrá elegir el modo con el que desea comunicarse con esas FC que tenga en la CWP, pudiendo elegir entre transmisión y recepción o solo recepción. Al seleccionar el nuevo modo, la CWP envía un *re-INVITE* actualizado el estado *Idle* al modo elegido hacia el SERCOMAV, el cual reenvía a la radio IP simulada en el RIS.

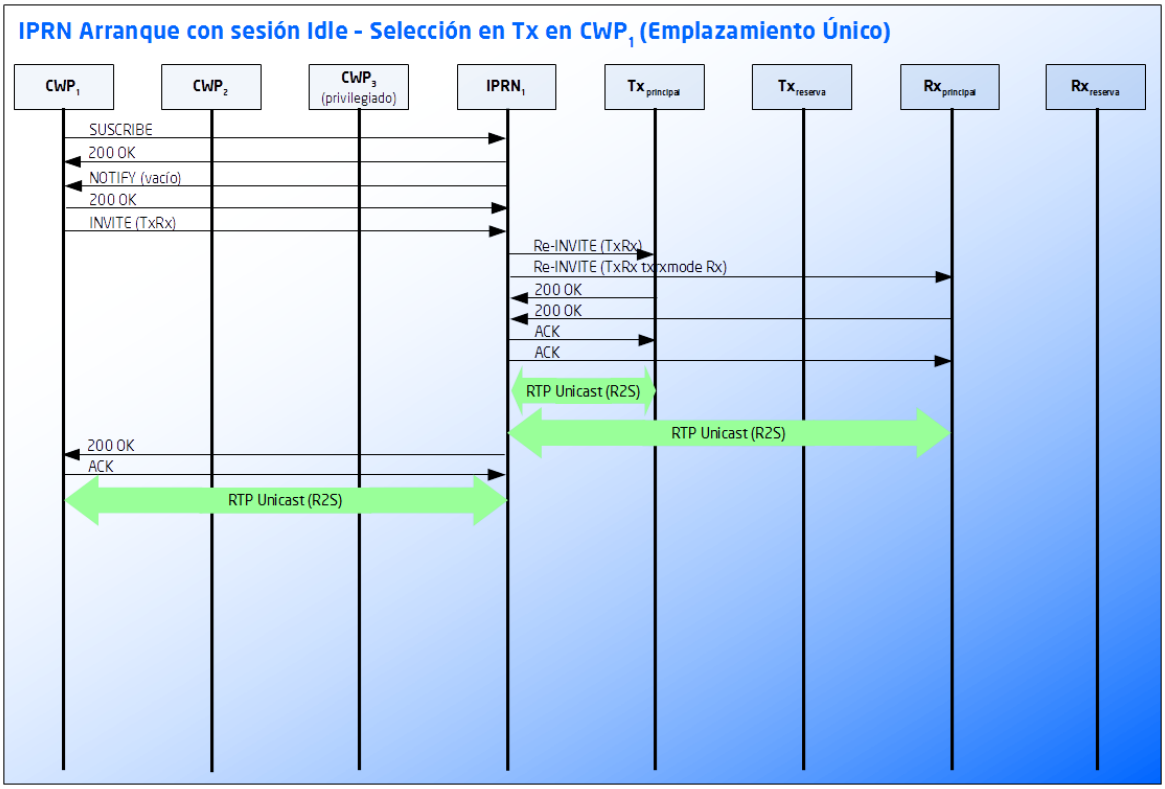

Figura 7: Cambio en la sesión SIP al seleccionar una FC en TxRx

Cuando tengamos dicha modificación, es cuándo podremos realizar una comunicación por voz. Dicha comunicación se produce por el protocolo RTP, el cual se explica a continuación.

2.3.2 Real-time Transport Protocol (RTP)

Este protocolo define un formato de paquete estándar para el envío de audio y video [5]. Utilizado en los sistemas de comunicación que involucran medios de transmisión, como telefonía y radio, basado en funcionalidad *Push-To-Talk* (PTT). La mensajería RTP suele usar UDP como la mayoría de los protocolos de tiempo real. La estructura del RTP contiene campos de número de secuencia, lo que permite el seguimiento de una comunicación activa y, a las máquinas,

identificar correctamente los paquetes de audio. Aquí podemos ver una cabecera RTP y la explicación de cada uno de sus campos.

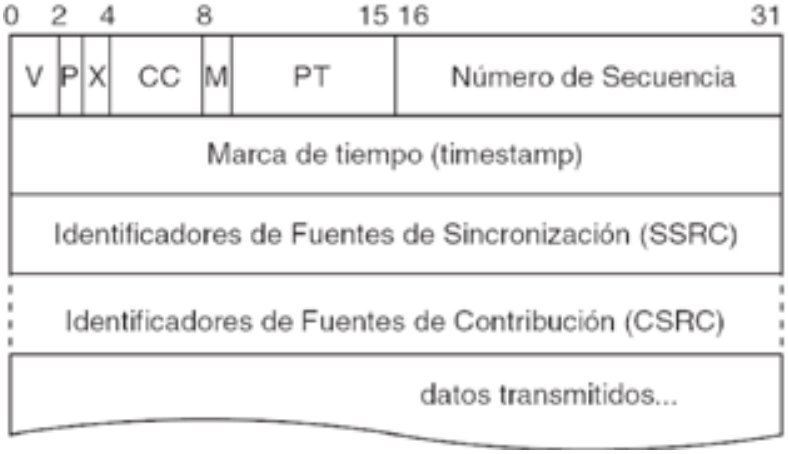

Figura 8: Cabecera de un mensaje RTP

- V: *Version*. Indica la versión que se está usando del protocolo RTP.
- P: *Padding* (Relleno). Rara vez usado, indica si los datos útiles llevan relleno para completar el tamaño del paquete
- X: *Extension*. Si está "*enabled*" entonces la cabecera RTP viene seguida de una cabecera adicional de extensión.
- CC: Contador CSRC. Usado por el receptor para saber la cantidad de *streams* RTP recibidos. En VoIP suele ser 0.
- M: *Marker*. Utilizado para eventos como el DTMF (*Dual-Tone Multi-Frecuency*), volúmenes y otros factores que puedan afectar a la carga útil del RTP
- PT: *Payload Type*. Aquí se denota el tipo de carga útil, como puede ser el códec.
- Número de secuencia. Lo usa el dispositivo receptor para saber si ha habido pérdida de paquetes durante la transmisión.
- Marca de tiempo. Usado para colocar el audio recibido en el orden correcto de sincronismo.
- SSRC. Identificación de la fuente. Suele dar valores aleatorios para evitar el mismo SSRC id en la misma retransmisión RTP.
- CSRC. Este campo no se usa en VoIP y, además, no es obligatorio para formar la cabecera.

Al ser la radio un sistema *Half-duplex*, en el que solo un usuario puede transmitir, es necesario añadir un sistema de permita al usuario ocupar la línea y que no pueda ser interrumpido, por lo que se utiliza la funcionalidad PTT y *Squelch* (SQ). La funcionalidad PTT posibilita indicar que un abonado está ocupando una línea de comunicación, por lo que imposibilita que otro abonado pueda ocuparla. Esta información es enviada por el mensaje RTP, que es quien maneja la mensajería de voz, y así asegurar que en una misma secuencia de RTP solo hay un PTT.

El SQ, como definición, es un umbral de detección de audio que asume que, si la retransmisión es inferior a dicho umbral, se trata de ruido e imposibilita la salida de audio y en cambio, si la supera, abre el circuito de comunicación y permite la recepción de dicho audio. Esta

funcionalidad, también transmitida en la mensajería RTP, permite a la radio indicar cuándo hay recepción de audio y cuándo no.

Con ambas funcionalidades activas, se puede informar al usuario del estado actual de la comunicación radio, como por ejemplo señalizándole cuando hay alguien ocupando esa línea (PTT de otro abonado), cuando tiene una recepción (SQ) o cuando él mismo está ocupando la línea (PTT propio).

En el caso particular de este proyecto, la mensajería RTP también se utiliza a modo de supervisión de presencias, utilizando su funcionalidad R2S. Esta mensajería se manda constantemente entre radio y servidor con unos tiempos de respuesta muy delimitados, lo que posibilita que, si una radio IP perdiese la conexión, la ausencia de mensajería R2S avisa al servidor de que algo no va correctamente y anula la sesión SIP que tuviese establecida en ese momento.

2.4 Radio IP Simulator (RIS)

Una vez conocido el SCV y los protocolos a utilizar, vamos a explicar el funcionamiento del RIS debe realizar para su correcto funcionamiento con el sistema.

El RIS debe recibir los archivos de configuración del TMCS para conocer las diferentes radios IP definidas. Como hemos visto en el apartado explicativo del TMCS, las radios simuladas por el RIS pueden ser tanto Rx, TxRx, Tx+Rx, Rx – Rx, TxRx – TxRx y Tx+Rx – Tx+Rx; las cuales el RIS debe ser capaz de diferenciar. Además, debe crear dichas radios con el nombre asignado en el campo de SIP-URI para que la comunicación SIP entre las máquinas sea correcta.

Una vez recibida la configuración desde el TMCS y generado las diferentes radios simuladas, el RIS debe lanzar una ventana HMI para que el usuario pueda manipular las radios con transmisiones de audio, comprobar el estado a tiempo real de las radios y simular el envío de mensajes SIP con determinadas características para ver cómo responde el SCV. Para ello, las radios creadas deben establecer su sesión SIP en modo Idle con el SERCOMAV y enviar la mensajería R2S para la correcta supervisión en el TMCS.

Con el sistema ya en ejecución, el RIS debe ser capaz de realizar cambio de sesión SIP establecida por cada radio, actualizar la sesión SIP si un controlador en la CWP selecciona una FC en alguno de los modos disponibles, señalizar el estado actual de la línea, hacer PTT para el envío de audio, poder enviar las distintas peticiones vistas en el protocolo SIP, introducir retardos en el envío de las presencias mediante R2S y más funcionalidades que explicaremos a continuación.

3 Desarrollo de la herramienta RIS

3.1 Introducción

En esta parte de la memoria vamos a explicar el desarrollo del RIS, así como las distintas herramientas que vamos a utilizar para dicho desarrollo. Al tratarse de un desarrollo realizado en Indra Sistemas S.A. hay ciertos elementos que no pueden ser explicados por estar ligados a la propiedad intelectual la empresa.

Para el desarrollo del RIS, se va a utilizar el lenguaje de programación C++ y el entorno de desarrollo llamado *Qt Creator* en un sistema operativo Linux, el cual explicaremos a continuación.

3.2 Entorno de desarrollo *Qt Creator*

Esta herramienta se trata de un *Integrated Development Environment* (IDE) multiplataforma programado en C++ con la idea de desarrollar aplicaciones con *Graphical User Interface* (GUI). Creado como un software libre y de código abierto, esta herramienta soporta los distintos sistemas operativos como GNU/Linux, Mac OS y Windows. Aunque inicialmente está pensado para utilizar el lenguaje C++ de forma nativa, también puede ser utilizado en otros lenguajes de programación a través de distintas librerías. En la siguiente imagen, vemos un ejemplo de cómo es este entorno de desarrollo.

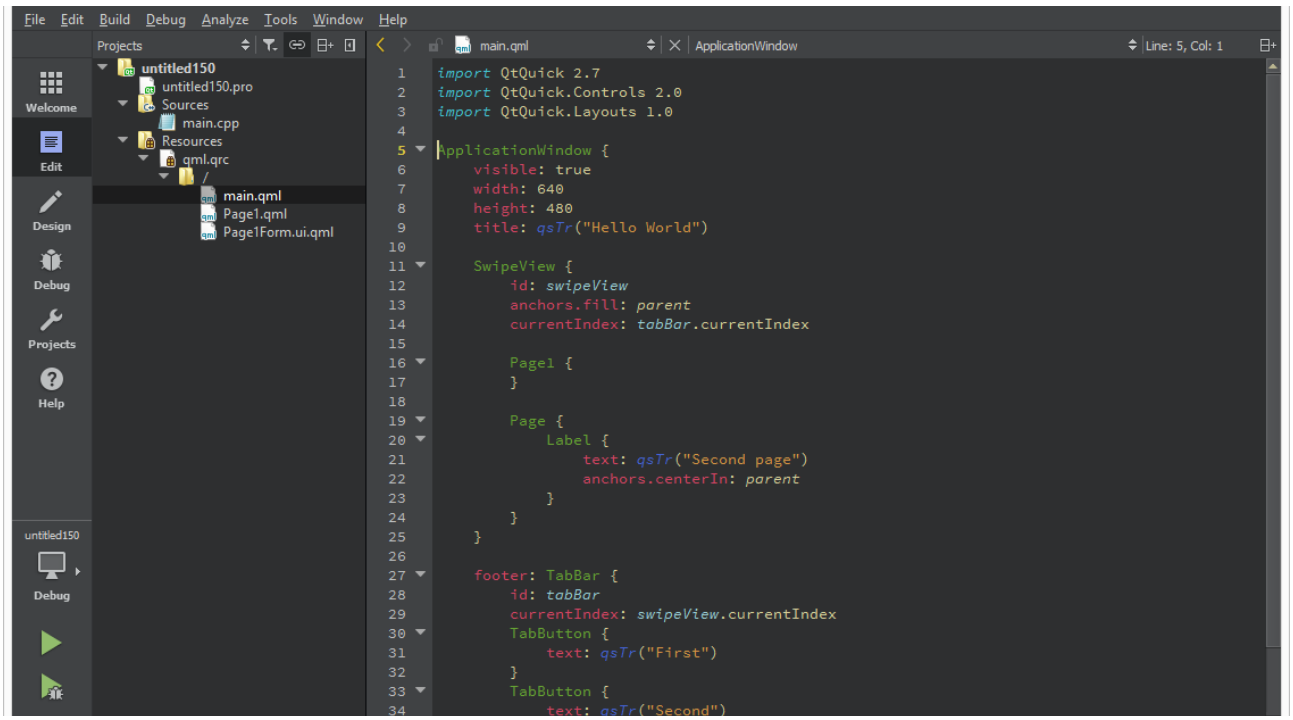

Figura 9: Entorno de desarrollo Qt Creator

Para nuestro desarrollo, vamos a utilizar el lenguaje C++ para crear todas las operativas y funcionalidades que el RIS pueda utilizar y, además, vamos a aprovechar las librerías de *Qt* para el diseño de la GUI de esta herramienta. El Garex300 utiliza el sistema operativo Centos 6.9 de 32 bits (Linux) para ejecutarse, por lo que vamos a instalar el programa *Qt Creator* en el mismo sistema operativo para que al finalizar el desarrollo, generar los archivos ejecutables y evitar posibles problemas de integración entre el código y el sistema operativo.

Además, el entorno de *Qt Creator* dispone de una pestaña de "Diseño". Esta pestaña nos va a resultar de gran utilidad a la hora de diseñar las ventanas que el RIS va a utilizar, ya que podemos ver en todo momento su resultado final antes de hacer una simulación. Nos permite añadir botones con el logo que deseemos, huecos para introducir texto, huecos para que introduzca texto el usuario de la herramienta, etc. Podemos ver en la siguiente figura una captura de cómo es la ventana de "Diseño" del programa *Qt Creator*.

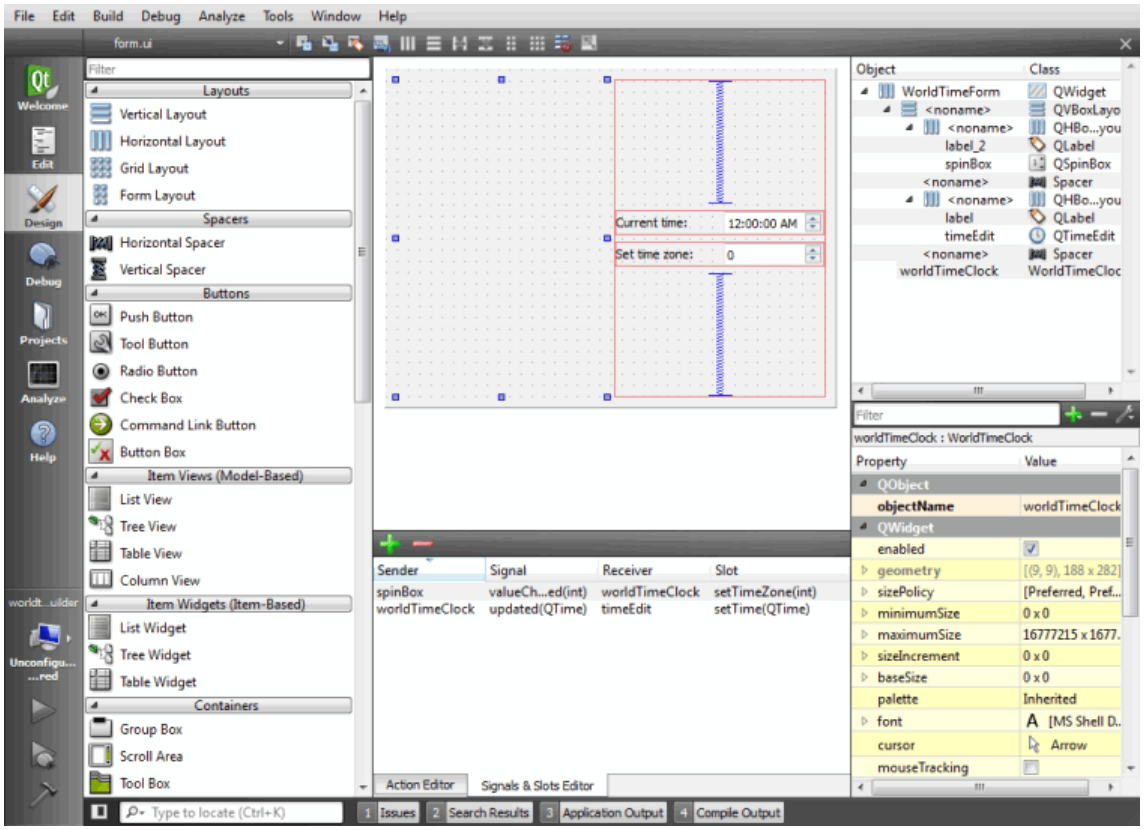

Figura 10: Ventana de diseño de Qt Creator

3.3 Estructura del RIS

Antes de empezar a escribir el código, hay que tener claro las funcionalidades y operativas que el RIS va a tener, por lo que vamos a explicar exhaustivamente todas las características de la herramienta y dejando a un lado, de momento, la parte de la interfaz gráfica de usuario.

Como hemos comentado en el apartado 2.4, el RIS recibe del TMCS los datos necesarios para la creación de las radios IP simuladas. Estos datos son entregados en varios archivos "`.txt`" donde se detalla la totalidad de la información declarada en la configuración hecha en el TMCS. Centrándonos en la parte de las uniones radio (UR), podemos obtener los siguientes campos:

- ID: Número que identifica a la unión radio dentro de los parámetros de la configuración.
- Nombre: Nombre que ha recibido la UR.
- Tipo de equipo: Si es digital o analógica. En nuestro caso, siempre serán digitales.
- Topología: Si se trata de un equipo único (TxRx), dos equipos (Tx+Rx), dos equipos con principal/reserva (TxRx, TxRx) o cuatro equipos (Tx+Rx, Tx+Rx).
- ID IPRN: Número que identifica el IPRN que va a gestionar el ER.
- SIP URI Tx principal: En caso de TxRx, debe ser igual que SIP URI Rx principal.
- SIP URI Rx principal: En caso de TxRx, debe ser igual que SIP URI Tx principal.
- SIP URI Tx reserva: Puede estar vacío si el ER no tiene Tx reserva.
- SIP URI Rx reserva: Puede estar vacío si el ER no tiene Rx reserva.
- Número máximo de sesiones SIP.
- Retardo del Tx principal.
- Retardo del Rx principal.
- Retardo del Tx reserva.
- Retardo del Rx reserva.
- Tiempo de cola de Squelch del Rx principal: Tiempo de remanencia desde que se deja de recibir el SQ hasta que desaparece en el mensaje RTP.
- Tiempo de cola de SQ del Rx reserva.
- Tiempo de reactivación de SQ del Rx principal: Tiempo que tarda en reactivar el campo de SQ del mensaje de RTP. Pensado para evitar falsas comunicaciones o rebotes de audio
- Tiempo de reactivación de SQ del Rx reserva.
- *Timeout* sin SQ: Tiempo máximo en el que se puede recibir el SQ. Si no llega dentro de este tiempo, se determina que no hay audio que recibir.
- Códecs: Posibles códecs que la UR puede utilizar.

Todos estos parámetros provienen de un fichero molde, que, dependiendo de la configuración, hay parámetros que no son enviados o simplemente se envían vacíos. Una vez sabemos cuáles son los datos recibidos por el TMCS, podemos generar los distintos códigos fuente para trabajar toda esa información.

Para esquematizar el desarrollo de la herramienta, vamos a hacer una diferenciación entre el funcionamiento interno de la herramienta (generación de radio según la configuración del TMCS, mantenimiento y/o modificación de las sesiones SIP, arranque de la aplicación, etc.) y entre la GUI (diseño de ventanas, texto mostrado en pantalla, resolución, etc.). Siguiendo un orden, empezaremos explicando el funcionamiento interno de la herramienta, comúnmente denominado *back-end*.

3.4 *Back-End* RIS

Al tratarse de una herramienta que va a tener un amplio desarrollo debido a la cantidad de características descritas anteriormente de la tecnología de RoIP, vamos a crear distintas cabeceras para facilitar el entendimiento del código fuente. Para llevar a cabo esta tarea, por cada código fuente descrito en el lenguaje C++ (`código.cpp`), escribiremos una cabecera donde definiremos las distintas estructuras o clases que podamos utilizar en nuestro código fuente (`código.h`). De esta manera, al iniciar el código fuente, añadimos una línea en la que incluyamos las distintas cabeceras que podamos necesitar y no tener que crear las distintas estructuras o clases descritas en dichas cabeceras una y otra vez por cada código fuente.

Como primer paso, desarrollaremos un código de inicio del RIS y la obtención de la información proporcionada por el TMCS. Después de eso, vamos a trabajar cada radio IP simulada desde tres enfoques distintos: sesión, frecuencia y radio. En la parte de sesión gestionaremos el control y mantenimiento de las posibles sesiones que los distintos ER puedan tener (enviar *re-INVITE*, mensajes de audio RTP, señalización de presencias, etc.), en la parte de la frecuencia enfocaremos los ER que trabajan dentro de una misma frecuencia como un mismo equipo (ER que deben transmitir, que audio se trasmite por cada frecuencia, generación de PTT, etc.) y, por último, la parte de radio, la cual se administrará las cualidades de cada ER en concreto (retardo añadido, sesiones máximas disponibles, respuestas concretas, etc.).

A continuación, se explicará detalladamente tanto la obtención de la información dada por el TMCS como cada uno de estos enfoques, adjuntando capturas de las líneas de código correspondiente. Como ya hemos comentado antes, hay mucha parte del código que interacciona con librerías o desarrollos hechos por Indra Sistemas S.A. que están protegidas por derechos de propiedad industrial, por lo cual no serán explicados en esta memoria.

3.4.1 Código de arranque

En este apartado se explica el proceso de arranque de la herramienta RIS. Para ello, se han añadido figuras con las partes del código más relevante para explicar dicho proceso y, al final, un diagrama de flujo explicando todo el proceso aquí descrito.

```
void RadioController::onStart(bool * ok)
{
 try
 {
  initializeCommunications();
  initializeAudio();
  initializeVCSMW();
  *ok=true;
 }
 catch(Exception& ex)
 {
  Utilities::printException(ex);
  *ok=false;
 }
}
```

Figura 11: Función `onStart`

Como vemos en la Figura 11: Función `onStart`, para arrancar el programa realizamos la llamada a tres funciones: "`initializeCommunications`", "`initializeAudio`" e "`initializeVCSMW`". Vamos a explicar las tres funciones en el orden que son llamadas en el código, por lo que empezaremos con "`initializeCommunications`".

```
void RadioController::initializeCommunications()
{
 SipAgent::Cfg sipAgentCfg;
 SipAgent::CfgTransport cfgTransport;
 QList<SipAgent::CfgTransport> cfgTransports;
 sipAgentCfg.MixerClock=0;
 // Recuperar el número máximo de llamadas posibles
 QString sMaxCalls = Utilities::getKeyValue("server", "maxcalls");
 bool ok;
 int iMaxCalls = sMaxCalls.toInt(&ok);
 if( ok && (iMaxCalls > 0) )
 {
  sipAgentCfg.MaxSimultaneusCalls=iMaxCalls;
  sipAgentCfg.MixerMaxSlots=iMaxCalls * 2;
 }
 else
 {
  sipAgentCfg.MaxSimultaneusCalls=900;
  sipAgentCfg.MixerMaxSlots=1300;
  sipAgentCfg.SupportedVersions.insert("phone.01");
  sipAgentCfg.SupportedVersions.insert("radio.01");
  sipAgentCfg.SupportedVersions.insert("radio.02");
  sipAgentCfg.SupportedVersions.insert("recorder.01");
  sipAgentCfg.SupportedVersions.insert("legacy-eu.01");
 }
 SipAgent::Init(sipAgentCfg);
 cfgTransport.Ip = Utilities::getKeyValue("server", "address");
 cfgTransports.append(cfgTransport);
 SipAgent::SetTransports(cfgTransports);
 connect(SipAgent::instance(), SIGNAL(incomingCall(InCallSg)),
this, SLOT(onIncomingCall(InCallSg)));
 connect(SipAgent::instance(),
SIGNAL(newWg67Subscription(InSubsSg)), this,
SLOT(onNewWg67Subscription(InSubsSg)));
}
```

Figura 12: Función `initializeCommunications`

En este código, se va a iniciar las comunicaciones con el resto de las máquinas del sistema. Concretamente, iniciaremos las comunicaciones WG-67 (*Working Group 67*) de la normativa ED-137 de EUROCAE e iniciaremos las comunicaciones SIP. El estándar WG-67[6] de la normativa ED-137 de EUROCAE recoge las funcionalidades y servicios obligatorios para las comunicaciones en sistemas ATM (*Air Traffic Management*). Todo SCV (Sistema de Comunicación por Voz) que vaya a ser utilizado en sistemas ATM debe cumplir con este estándar.

Siguiendo con las líneas del código, primero debemos obtener cierta información sobre la configuración del servidor SIP, que en nuestro caso será el SERCOMAV. Concretamente, queremos conocer el número máximo de sesiones SIP que puede establecer y el número máximo de radios que puede gestionar el SERCOMAV. Para obtener esta información, se llama a la función "`getKeyValue`" del código "`Utilities.cpp`".

```
QString Utilities::getKeyValue(QString group, QString key)
{
 QSettings settings(CFG_FILE, QSettings::IniFormat);
 settings.beginGroup(group);
 QString value = (settings.value(key,"")).toString();
 settings.endGroup();
 return value;
}
```

Figura 13: Función `getKeyValue`

Esta función busca en los archivos de configuración obtenidos inicialmente, en la sección descrita en el primer elemento introducido en la función, el valor del campo del segundo elemento introducido en la función. En el caso en el que estamos, busca en el archivo de configuración, concretamente en la zona "*server*" el valor del campo "*maxcalls*" y devuelve el valor de dicho campo. Así, si en la configuración se ha editado o añadido un nuevo valor al de por defecto (900), lo usamos para la simulación. Si el valor que devuelve la función "`getKeyValue`" es 0, se añaden unos los valores por defecto que se pueden observar en la figura Figura 12 dentro del corchete del "*else*".

Una vez obtenidos o añadidos por defecto estos valores, volvemos a usar la función "`getKeyValue`" para obtener la IP del servidor y así iniciar las comunicaciones SIP y WG-67. Para ello, utilizamos las librerías de SIP y dos funciones: "`onIncomingCall`" y "`onNewWg67Subscription`". La primera función, "`onIncomingCall`", teniendo en cuenta que la llamada SIP que se va a realizar es de tipo radio, modificamos la URI-SIP que está normalizada para que se envié correctamente en el paquete del *INVITE* y obtenemos los códecs que han sido añadidos en la configuración. Si no se han añadido códecs especiales, se añaden los que el Garex300 acepta por defecto: PCMU[7], PCMA[7], G729A[8], G729B[8] y G728[9]. Una vez tengamos esta información, actualizamos la lista de códecs que van incluidas en el mensaje y mandamos la respuesta al *INVITE* que recibimos del SERCOMAV. Si al acceder a la información de la radio no se ha encontrado nada, se manda un mensaje de respuesta de declinación y así el SERCOMAV da la radio por no disponible.

```
void RadioController::onIncomingCall(InCallSg data)
{// NUEVA LLAMADA DE RADIO
 if (data->Type == SipCallType::Rd)
 {
  data->Local = data->Local.mid(data->Local.indexOf("<"));
  if(radios.contains(data->Local))
  {
   QString uri = radios[data->Local]->getUriNormalized();

uri=uri.replace("/","").replace("<","").replace(">","").replace("@",
"_").replace("sip:","");
   QStringList codecs;
   QString codecList =
Utilities::getKeyValue("codecPreferences",uri);
   if(codecList!= 0 )
   {
    codecs = codecList.split("/");
   }
   else
   {
    for(int i = 0; i<6 ; i++)
    {
     std::stringstream str;
     str << i+1 ;
     QString aux =
Utilities::getKeyValue("defaultCodecs",str.str().c_str());
     if(aux.compare("PCMU")==0 || aux.compare("PCMA")==0 ||
aux.compare("G729A")==0 || aux.compare("G729B")==0 ||
aux.compare("G728")==0 || aux.compare("R2S")==0 )
     {
      codecs <<
Utilities::getKeyValue("defaultCodecs",str.str().c_str());
     }
    }
   }
   radios[data->Local]->setCodecs(codecs);
   radios[data->Local]->invite(data);
  }
  else
  {
   qWarning() << "RadioDevice::onIncomingCall No hay recurso radio
para la URI:" << data->Local << " (503 Service Unavailable)";
   try
   {
    RdSipCall::answer(data->CallId, 503);
   }
   catch(coresip::Exception& ex)
   {
    Utilities::printException(ex);
   }
  }
 }
```

Figura 14: Función `onIncomingCall`

En la segunda función "`onNewWg67Subscription`", teniendo en cuenta que la información de la radio está correcta, se subscribe la radio al servidor de WG-67. Si no, se manda un mensaje para cancelar el mensaje recibido de subscripción.

```
void RadioController::onNewWg67Subscription(InSubsSg data)
{
 if(radios.contains(data->Local))
 {
  radios[data->Local]->onNewWg67Subscription(data);
 }
 else
 {
  Wg67EvServer::rejectSubscription(data);
 }
}
```

Figura 15: Función `onNewWg57Subscription`

Continuando con el código de la Figura 11: Función `onStart`, se realiza una llamada a la función "`initializeAudio`". En esta función, creamos un mixer para que la radio sea capaz de reproducir y/o modificar los archivos de audio que la herramienta RIS dispone.

```
void RadioController::initializeAudio()
{
 try
 {
  mixer = Mixer::create();
 }
 catch(Exception& ex)
 {
  Utilities::printException(ex);
 }
}
```

Figura 16: Función `initializeAudio`

Y, por último, siguiendo el código de la Figura 11, se llama a la función "`initializeVCSMW`".

```
void RadioController::initializeVCSMW()
{
 InitConfig cfg;
 bool ok;
 cfg.operationErrorInit = ConfigurationSystem::emergency;
cfg.numAttempt =
Utilities::getKeyValue("ConfigSystem","ReTrying").toInt(&ok,10);
 cfg.tmcsTimeout =
Utilities::getKeyValue("ConfigSystem","Timeout").toInt(&ok,10);
 cfg.tmcsDataTimeout =
Utilities::getKeyValue("ConfigSystem","DataTimeout").toInt(&ok,10);
 cfg.deviceInfo.DeviceType = SCVSystem::position;
 strcpy(cfg.myIP,
Utilities::getKeyValue("Server","address").toStdString().c_str());
 cfg.listenPort =
Utilities::getKeyValue("ConfigSystem","ListenPort").toInt(&ok,10);
 cfg.destPort =
Utilities::getKeyValue("ConfigSystem","RemotePort").toInt(&ok,10);
 strcpy(cfg.actualFilesPath,
Utilities::getKeyValue("ConfigSystem","CurrentCfgPath").toStdString().c_str());
 strcpy(cfg.emergencyFilesPath,
Utilities::getKeyValue("ConfigSystem","EmergCfgPath").toStdString().c_str());
 strcpy(cfg.tmpPath,
Utilities::getKeyValue("ConfigSystem","TmpCfgPath").toStdString().c_str());
 cfg.multiplierTimeout =
Utilities::getKeyValue("ConfigSystem","MultiplierTimeout").toInt(&ok,10);
 int resCfg = StartConfigSystem(cfg, incomingCfg, this, false, false, false);
 if (resCfg != CONFIG_SYSTEM_OK)
 {
  qWarning() << "Initialize VCSMW failed. Check if they are minimum TMCS
configuration in ./dat/emergency";
  exit(0);
 }
 loadConfiguration(&cfg);
}
```

Figura 17: Función `initializeVCSMW`

En esta función obtenemos todos los parámetros necesarios del *middleware* (MW) del SCV para el correcto funcionamiento del sistema de presencias que el TMCS tiene sobre el resto de los elementos del sistema. Primero obtenemos los datos de configuración necesarios para el arranque, como pueden ser el tiempo configurado entre *re-INVITES*, la dirección IP que va a tomar, el puerto para la comunicación SIP o el *timeout* configurado en cada ER. Una vez comprobado que todos los datos se han obtenido correctamente, llamamos a la función "`loadConfiguration`".

La función "`loadConfiguration`" primero realiza un chequeo para comprobar que el formato del archivo de configuración es correcto. Una vez corroborado que el formato está bien, procedemos a obtener la información correspondiente de los ER que vamos a simular con el RIS. Para ello, vamos a clasificar la información las FC configuradas, ya que cada FC tiene asignada una o varias UR y estas, a su vez, están compuestas por los distintos ER, que pueden ser Tx, Rx o TxRx. Para ello se diseñan dos bucles "*for*" que hacen referencia a esta clasificación, siendo el primero el encargado de separar por FC y el segundo por UR. Se diseña un tercer bucle donde se tiene en cuenta si el ER tiene la función *Climax* habilitada o no. Esta función característica de las radios sobre IP calcula el retardo interno del sistema entre cada ER y la CWP y aplica ese retardo

calculado en las transmisiones que haga. Esta funcionalidad es exclusiva y de uso obligatorio de los ER que estén dentro de FC con dos o más UR configuradas.

En definitiva, con estos bucles podemos organizar la información y saber qué tipo de ER es (TxRx, Tx+Rx, Rx, TxRx - TxRx, Tx+Rx – Tx+Rx o Rx – Rx), a que UR pertenece, a que FC pertenecen estas UR y si hay más UR en esa misma FC. De esta forma de organización, nos facilita la posterior representación de la información en el interfaz de usuario.

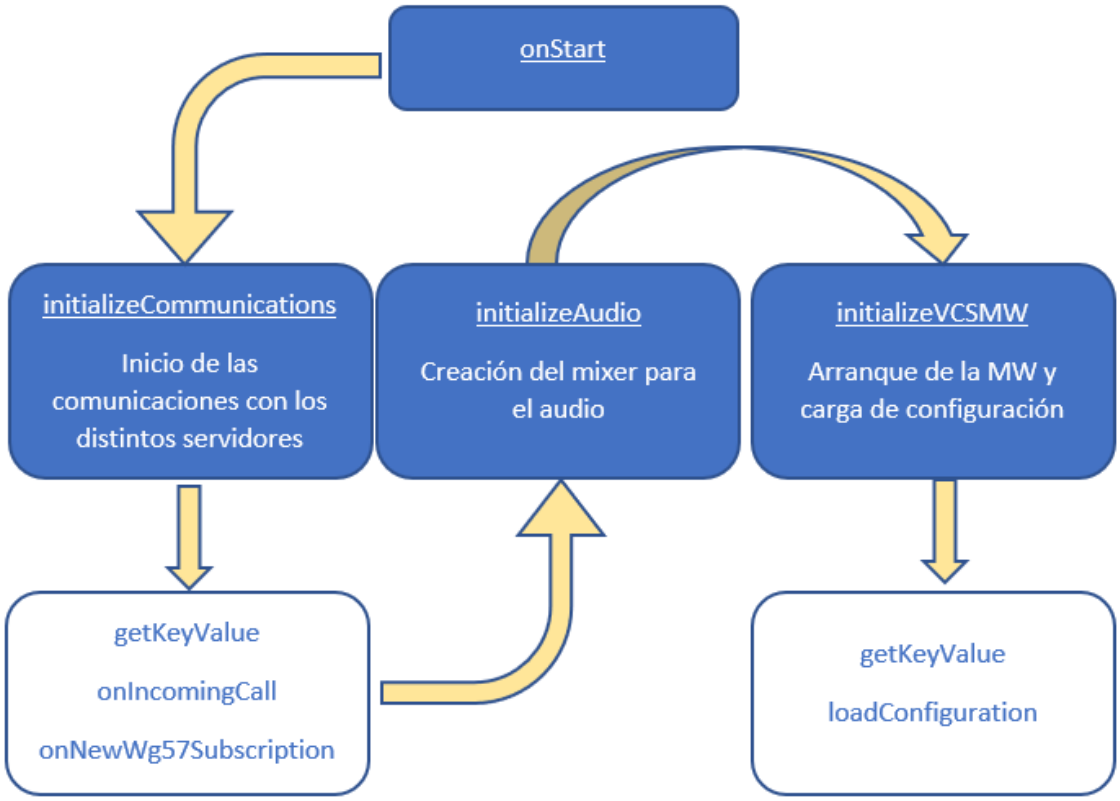

Figura 18: Diagrama de flujo del arranque del RIS en el código `RadioController.cpp`

Una vez finalizado esta sucesión de procesos, se tiene la información necesaria para ejecutar el proceso de arranque de la herramienta. Para ello, disponemos del siguiente código fuente.

```
int main(int argc, char *argv[])
{
 QApplication a(argc, argv);
 RadioController c;
 int iReturn = -1;
 if(c.start())
 {
  qDebug() << "RIS started...";
  iReturn = a.exec();
  qDebug() << "Ending application...";
 }
 else
 {
  qDebug() << "Radio controller error. Exit application";
 }
 return (iReturn);
}
```

Figura 19: Código de arranque

Se puede contemplar cómo se reciben los parámetros del código fuente "*RadioController.cpp*" y se lanza el arranque de la herramienta. Si existiese algún error durante la ejecución, sería mostrado por pantalla para que el usuario lo conozca. También se tiene en cuenta si hay algún error durante el proceso de obtención de información por parte del código fuente "*RadioController.cpp*", lo que provoca que el arranque de la herramienta se detuviera.

3.4.2 Funcionalidades y operativas durante la ejecución.

Una vez hemos diseñado el proceso de arranque, hay que tener en cuenta los distintos escenarios que nos podemos encontrar durante la ejecución del RIS y las distintas funcionalidades que la tecnología RoIP nos ofrece. Para ello, en este apartado se procede a explicar el código fuente que va a permitir al RIS disponer de esa operativa.

Como se ha dicho anteriormente, hemos enfocado la herramienta en tres puntos de vista: frecuencia, radio y sesión. Se han desarrollado tres códigos distintos (`Frecuncy.cpp`, `RadioIP.cpp` y `Session.cpp`) que recogen las distintas operativas que van a tener durante la ejecución del RIS.

3.4.2.1 Funcionalidades de un equipo radio en el RIS

Este punto de vista, desarrollado en el código fuente `RadioIP.cpp`, se recaba toda la información sobre las características que dispone cada ER como su funcionamiento en los distintos escenarios posibles que pueda encontrarse dicho ER. El siguiente listado se muestra toda la operativa que se le ha dado a este código fuente:

- FC a la que pertenece este ER.
- Tipo de ER que se trata (Rx, Tx o TxRx).
- SIP-URI que identifica a cada ER.
- Obtener y modificar el retraso en la recepción. Este parámetro solo lo tendrán los ER que sean Rx o TxRx. Se configura para que, en caso de tener varios ER Rx, se produzca la recepción de manera simultánea.
- Obtener y modificar el número máximo de sesiones. Se puede configurar un número máximo de sesiones que puede tener establecido un ER de manera simultánea, por lo que, si se llegase a alcanzar esta cifra, las nuevas sesiones que se quieran establecer contra el ER serán rechazadas. Se utiliza para limitar el uso de ciertos ER para unos usuarios específicos.
- Obtener datos característicos de la transmisión. Estos datos pueden ser si se tiene habilitado el SQ o si hay confirmación de PTT (Rx envía la frecuencia portadora al Tx para confirmar la correcta recepción de la transmisión).
- Estado actual del ER. Comprobaciones de si ER se encuentra encendido o si sigue mandando la mensajería con presencias R2S (*KeepAlive*).
- Modificar el estado del ER. El usuario tiene la posibilidad de apagar el ER, deshabilitarlo o encenderlo. Por ello, hay que tener en cuenta en qué estado nos encontramos y cuál ha sido la acción realizada por el usuario, para actualizar el estado de la radio conforme a la acción desempeñada por el usuario.

- Modificaciones del estado de ER cuando se realice una transmisión. Si el ER con el que trabajamos es un Tx o TxRx, se tiene que modificar la cabecera de RTP para el caso de transmisión, el *INVITE* de la mensajería SIP debe ser de tipo TxRx y actualizar su estado de supervisión en el RIS (realizando una comunicación).
- Modificaciones del estado de ER cuando se recibe una transmisión. Si el ER con el que trabajamos es un Rx o TxRx, se tiene que modificar las distintas cabeceras del mensaje RTP, indicar que se está recibiendo una comunicación en la supervisión del RIS y permitir la reproducción del audio recibido.

Todas estas operativas son necesarias para que cada ER funcione correctamente por separado, sin tener en cuenta el papel que desempeñan dentro de cada FC.

3.4.2.2 Operativa de la sesión en el RIS.

Una vez vista las distintas funcionalidades y operativas que tiene el RIS por cada ER, debemos tener en cuenta que hay que establecer correctamente sesiones con el resto de los elementos del sistema, como son el SERCOMAV o la CWP. Por ello, en el código `Session.cpp` se recoge las siguientes operativas:

- Establecer sesión con el SERCOMAV. Por defecto, cuando se arranque el RIS, se debe enviar un *INVITE* de tipo *Idle* por cada ER que se vaya a simular en el RIS.
- Sesiones de supervisión. Una vez establecidas estas sesiones SIP, el RIS debe enviar la mensajería de RTP (R2S) cada cierto tiempo para que el SERCOMAV sepa que sigue estando operativa.
- Modificación sesión SIP. El RIS es capaz de modificar las sesiones SIP que se establecen, por lo que es capaz de enviar *re-INVITEs* del tipo que el usuario haya elegido (Rx, TxRx o *Coupling*).
- Establecimiento de nuevas sesiones SIP. Una CWP debe ser capaz de conectarse aun ER simulado por el RIS, en el modo que se haya seleccionado en la CWP, siempre y cuando no se haya superado el número máximo de sesiones disponibles o el usuario indique que se rechacen las nuevas sesiones.
- Envío de respuestas automáticas. Por defecto, ante peticiones de nuevas sesiones, se envía la respuesta 200 *OK*. En cambio, si se ha superado el número máximo de sesiones disponibles, el RIS debe ser capaz de responder correctamente y rechazar esa invitación.
- Envío de respuestas seleccionadas. El usuario es capaz de configurar en el RIS la respuesta que quiere que se envíe en caso de recibir una invitación para establecer una nueva sesión, por lo que el RIS debe ser capaz de responder correctamente con la respuesta elegida por el usuario.
- Cancelación de sesiones. El RIS es capaz de cancelar el mismo las sesiones que tenga establecidas enviando un *BYE* al otro elemento del sistema debido a que el usuario ha decidido eliminar esa sesión o ha apagado el ER que tenía esa sesión SIP establecida.
- Cancelación ajena de sesiones. El RIS debe ser capaz de contestar correctamente ante la petición de cancelación de una sesión SIP ya establecida por parte de los elementos externos del sistema.

- Mensajería de transmisiones. Al realizar o recibir una transmisión, se debe enviar la mensajería RTP correcta en cada caso, actualizando las cabeceras de RTP según la ocasión.

La mayoría de las veces que este código entre en funcionamiento será en el proceso de arranque para establecer las sesiones SIP iniciales contra el SERCOMAV y el continuo envío de mensajes para indicar que sigue operativo. El resto de los casos se deben tener en cuenta ya que el usuario puede querer modificar sesiones ya establecidas o un elemento externo del sistema, como la CWP, puede reconfigurar dichas sesiones.

3.4.2.3 Funcionalidades de una frecuencia en el RIS.

Las funcionalidades de una frecuencia pueden considerarse como la capa más alta de todas las funcionalidades que dispone el RIS, ya que toda acción o modificación llevada a cabo contra una FC repercutirá sobre los ER que componen esa FC y sus respectivas sesiones establecidas. Las funcionalidades de una FC son las siguientes:

- Número de frecuencia. Se debe obtener el número de la frecuencia que se quiere simular como por ejemplo el número 118.700.
- ER que componen dicha FC. Se debe clasificar y unificar cada ER en función de la FC con la que trabajen.
- Estado actual de los ER que componen la FC. Conocer el estado actual de todos los ER que componen la FC para saber por cuál de ellas se puede transmitir o recibir las comunicaciones.
- Modificar el estado de los ER. Si se quiere realizar un apagado de todos los ER pertenecientes a la FC, o se quiere realizar una transmisión o se produce una recepción por dicha FC, esta debe ser capaz de modificar los ER que la componen de acuerdo con cada escenario que se presente.
- Obtener y modificar características de la FC. Deben aplicarse las características configuradas en el TMCS para cada FC que se simula y poder modificarlas en el RIS. Estas características pueden ser el retardo aplicado a la funcionalidad *Climax* o si tiene un identificador de grupo de BSS.

Al tratarse de la capa más alta de funcionalidades, la mayoría de las funcionalidades que se quieran hacer vienen recogidas sobre todo en el código de `RadioIP.cpp`, por lo que en el código `Frecuncy.cpp` se tiene en cuenta que ER forman parte de cada FC para así aplicar dichas funcionalidades a todos los ER pertenecientes a esa FC.

Todas las funcionalidades que se recogen en este apartado se explicarán con más detalle cuando se explique el interfaz de la herramienta, ya que será más fácil a la hora de entender dichas funcionalidades desde la operativa.

3.4.3 Diseño y desarrollo de la GUI

Como hemos explicado antes, *Qt Creator* dispone de una pestaña de edición específica para interfaces de usuario. Al crear un nuevo proyecto, no genera tres archivos distintos: una cabecera para declarar las funciones (`nombrecodigo.h`), un código fuente (`nombrecodigo.cpp`) y un código de formas (`nombrecodigo.ui`). En este último recoge el diseño que hagamos en la pestaña de edición, como puede ser el añadir botones, posición en la ventana, texto introducido o celdas vacías, entre otras muchas cosas. Una vez hecho este desarrollo, con el código fuente se puede editar e implementar la distinta operativa que queremos que se haga. Por ejemplo, si añadimos un botón, se crea este objeto en el código fuente correspondiente y, en este código fuente, podemos indicar que funcionalidad tenga ese botón.

Este método de trabajo que implementa la herramienta *Qt Creator* facilita el diseño de interfaces gráficas y su implementación con las funciones que hemos desarrollado en los apartados anteriores.

Para la visualización del RIS se ha determinado una ventana emergente compuesta por: una barra lateral donde se indique la información básica de cada FC, indicando su número de frecuencia y elementos de supervisión de dicha FC; y una ventana con la información detallada de cada ER perteneciente a dicha FC, como puede ser la SIP-URI, tipo de sesión establecida o tipo de ER, entre otras opciones.

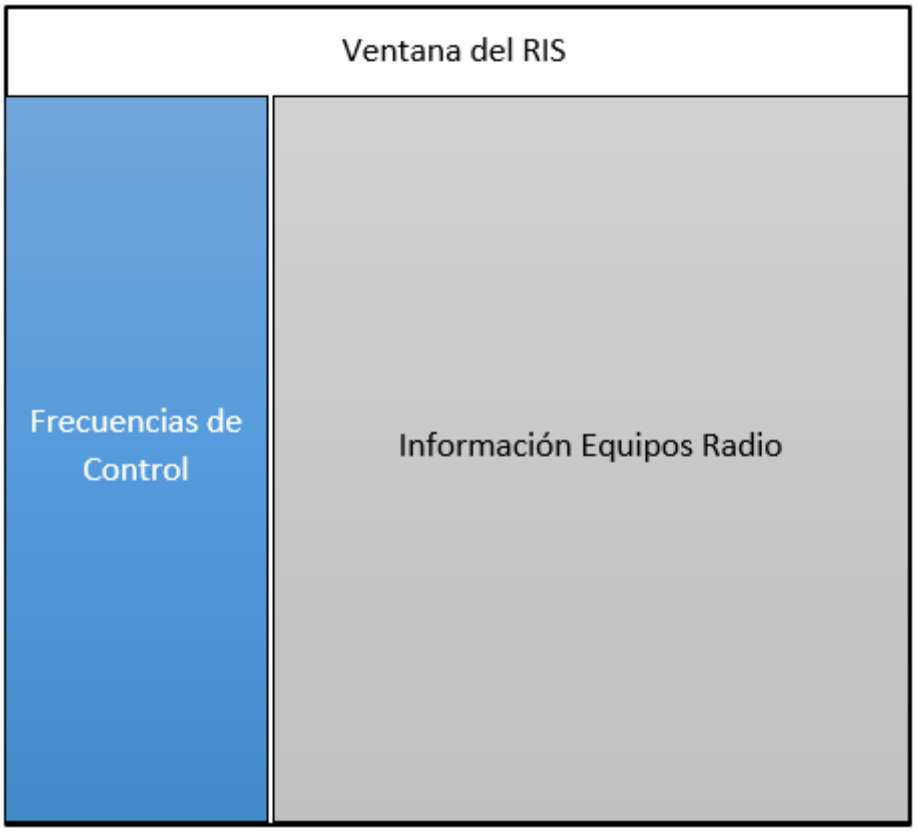

Figura 20: Esquemático de la presentación del RIS

Partiendo del esquema mostrado en la **¡Error! No se encuentra el origen de la referencia.**, se implementan tres códigos fuente con su respectivo código de formas por cada ventana y sus cabeceras. Los códigos "`RadioIPServerSimulatorApp.cpp`" , "`RadioIPServerSimulatorApp.ui`" y "`RadioIPServerSimulatorApp.h`" serán los responsables de la gestión de la ventana principal del RIS. Los programas "`ViewFrequency.cpp`", "`ViewFrequency.ui`" y "`ViewFrequency.h`" se encargan de la parte de las FC. Por último, "`ViewRadioIP.cpp`", "`ViewRadioIP.ui`" y "`ViewRadioIP.h`" gestionan la ventana con toda la información correspondiente a los ER. Al igual que en el desarrollo del código de arranque anteriormente explicado, también se ha diseñado un código fuente denominado "`ViewHelper.cpp`" para integrar en él funciones que vayan a necesitar el resto de los códigos. A continuación, se procede a explicar estos tres códigos fuente y su operativa.

3.4.3.1 Diseño de la ventana ER

En esta ventana se quiere diseñar con la finalidad de mostrar la supervisión en todo momento de cada ER, como puede ser el estado de la sesión SIP o características configuradas especialmente a ese ER, y ciertas funcionalidades para poder modificar tanto la sesión SIP como las características del ER.

Las funcionalidades y los elementos de supervisión que se han elegido son los siguientes:

- Apagado o encendido el ER.
- Suprimir la mensajería R2S que sirve para informar de que el ER sigue operativo (*KeepAlive*).
- Deshabilitar la opción de envío de SQ, para simular el caso de transmisión sin audio.
- Deshabilitar la opción de confirmación de PTT, para probar cómo se comporta el SCV en ese escenario.
- Eliminar las sesiones SIP que tenga establecida dicho ER eligiendo el tipo de mensaje SIP *BYE* que el usuario desee.
- Indicadores led por si la radio está transmitiendo o recibiendo.
- Editar el campo BSS para radios que tengan esta funcionalidad y sean Rx o TxRx.
- Añadir o eliminar el retardo en ER Rx o TxRx.
- Modificar el número máximo de sesiones que permite el ER.
- Añadir o eliminar el tiempo de retardo de la funcionalidad *Climax* en ER del tipo Tx o TxRx.
- Elegir un audio predeterminado para transmitir por radios del tipo Tx o TxRx.
- Rechazar las sesiones SIP nuevas que se quieran establecer contra el ER con posibilidad de elegir un motivo concreto.
- Topología del ER
- SIP-URI del ER
- Pestañas que muestran las opciones elegidas con respecto a los campos de BSS, retardo de Rx, número máximo de sesiones, audio elegido y motivo del rechazo de nuevas sesiones SIP.

Además, también se añade una parte de supervisión del estado actual de la o las sesiones SIP que tenga establecidas dicho ER, donde podremos observar los siguientes campos:

- Rx PTT-id. Número que identifica el PTT que se está recibiendo.
- Rx PTT. Si el Rx está recibiendo una transmisión.
- Rx *Mute*. Si la transmisión recibida por el Rx tiene deshabilitado la funcionalidad de SQ.
- Rx CLD. El tiempo que tiene de retardo en la función *Climax* el receptor, expresado en milisegundos.
- Tx PTT-id. Número que identifica el PTT del Tx.
- Tx PTT. Si el Tx está realizando una transmisión.
- Tx *Mute*. Si la transmisión realizada por el Tx tiene deshabilitado la funcionalidad de SQ.
- SQL. Muestra si la comunicación que hay establecida tiene SQ o no.
- *Type*. Tipo de sesión SIP que hay establecida.
- URI. SIP-URI del elemento que ha establecido la sesión SIP.

Con estas funcionalidades y elementos de supervisión, se quiere dar la posibilidad al usuario de modificar todo lo posible el ER y poder ver en todo momento como se comporta dicho ER. Todas las funcionalidades y elementos de supervisión nombrados son dependientes del ER que forme, por lo que solo los TxRx dispondrán de todas las opciones y los Tx o los Rx solo tendrán activas aquellas funcionalidades que le apliquen a los ER de su topología.

Esta ventana viene condicionada por la FC que se haya seleccionado en la ventana de FC, ya que, en función de esa selección, se mostrará los ER pertenecientes a la misma. En los casos en los que haya varios ER, en la ventana aparecerán barras de desplazamiento, tanto vertical como horizontal, para poder ver todos los ER.

Se debe tener en cuenta que los elementos de supervisión pueden variar debido a modificaciones que hagan tanto la CWP como el SERCOMAV, por lo que dichos campos también deben actualizarse cuando se produzcan cambios en la sesión SIP. Para ello, en el código fuente "`ViewRadioIP.cpp`" se han añadido funciones para actualizar los parámetros visibles en función a las variaciones realizadas en la sesión o sesiones SIP establecidas contra dicho ER.

Las funcionalidades que sean del tipo habilitar o deshabilitar, se van a implementar como botones, mientras que las funcionalidades en las que se tenga que elegir una opción entre varias posibilidades irá implementada en un menú desplegable con todas las opciones posibles.

3.4.3.2 Diseño ventana FC

El diseño de esta ventana va incorporado en el código fuente "`ViewFrequency.cpp`" que hemos mencionado antes. Para esta ventana se ha pensado en una lista en horizontal donde pueda verse todas las FC simuladas por el RIS. En caso de superar el tamaño de la ventana, aparecerán botones, tanto en la parte superior cuando no estemos en el inicio de la tabla como en la parte inferior cuando no estemos al final de la lista, para subir o bajar en el listado de FC.

En esta ventana, como en la ventana de ER, también se van a añadir distintas funcionalidades y elementos de supervisión. Tanto estas funcionalidades como los elementos de supervisión aplicarán sobre todos el ER pertenecientes a la FC que esté seleccionada. Esos elementos son los siguientes:

- Número elegido en configuración para definir esa FC.
- Apagar o encender todos los ER de la FC.
- Led indicativo del estado de Rx.
- Led indicativo del estado de Tx.
- Habilitar o deshabilitar el retorno de portadora. Esta funcionalidad sirve para que el Rx, al recibir una transmisión, envíe la señal portadora al transmisor para verificar que el mensaje ha llegado al destino.
- PTT ajeno para simular la recepción de una transmisión externa al sistema, como puede ser una transmisión proveniente de un avión.
- Elegir que audio se quiere transmitir por ese PTT ajeno, que será recibido por el o los ER Rx.
- Mostrar el audio que se ha elegido en la funcionalidad anterior.
- Habilitar o deshabilitar la detección de audio. Si está función está habilitada, solo aparecerá la señal de audio por altavoces cuando se transmitan sonidos y desactiva los altavoces en los silencios producidos en la transmisión.

Como en la ventana de ER, las funcionalidades que sean de habilitar o deshabilitar, son implementadas en botones con dos estados, mientras que la funcionalidad de elegir el audio será un botón en la que aparecerá un desplegable con las opciones posibles.

De elemento de supervisión tenemos los leds que indican la transmisión y la recepción. Estos leds van unidos con los leds que se muestran en el ER, por lo que si alguno de ellos se enciende (ya sea el del Tx o el del Rx) debe encenderse el led correspondiente de la ventana de FC.

Al hacer clic sobre la FC en el listado, este debe actualizar la ventana de ER mostrando únicamente los equipos que forman parte de la FC seleccionada. Además, para que el usuario sepa en que FC se encuentra, se ha añadido un cambio de color en la tecla de la FC seleccionada con respecto al color que tiene en reposo.

3.4.3.3 Diseño de la ventana del RIS

Para finalizar la parte del diseño de la GUI, se realiza una ventana que contendrá los dos apartados anteriores. El diseño y funciones de esta ventana es bastante más simple con respecto a las antes explicadas. En este caso, solo se diseña una ventana con las opciones de minimizar la ventana, aumentar o disminuir el tamaño y cerrar la aplicación.

La funcionalidad de aumentar el tamaño de la ventana también implica que aumente el tamaño de la ventana de ER y se pueda ver más opciones del listado de FC. El usuario puede pulsar el botón de aumento o disminución situado en la esquina superior derecha o editando dicho tamaño haciendo clic y arrastrando en cualquiera de los cuatro bordes de la ventana.

3.4.4 Compilación y ejecución del RIS

Tras desarrollarse todo el código fuente creado, tanto para el arranque del RIS y la operativa de las funcionalidades que esta herramienta posee como el código relacionado con la interfaz gráfica de usuario, se compila. El resultado de esta compilación nos da un archivo binario. Con la obtención de este archivo binario, podemos crear un script de arranque y así poder usar nuestra herramienta en cualquier máquina, ya que únicamente necesitaremos transportar el script, el binario y las librerías de las que dependa dicho binario.

4 Visualización y funcionamiento

4.1 Proceso de instalación y arranque

Como hemos comentado en el apartado 3.4.4, se ha diseñado un script como proceso de arranque. Para que el RIS tenga todos los elementos necesarios para su correcto funcionamiento, se ha generado un archivo comprimido (`Tool_RIS_Any_01_02.tar.gz`) con todos estos elementos necesarios para el arranque (binarios, librerías, archivos de audio, script de arranque y archivos de configuración).

Una vez tenemos el archivo comprimido en la máquina, que vayamos a utilizar como RIS, lo descomprimimos en el lugar deseado usando el siguiente comando, siempre como super usuario (*root*).

```
tar -zxvf Tool_RIS_Any_01_02.tar.gz /”ubicación deseada”/
```

Figura 21: Comando para descomprimir el RIS

Una vez hemos descomprimido el archivo, accedemos a la ubicación que hemos usado en el anterior comando y lanzamos el script de arranque. Para que funcione correctamente, se le debe añadir la dirección IP que tenga la máquina donde está instalada al comando para ejecutar el script de arranque.

```
cd / “ubicación deseada” /

./RIS_RIS_scr_unix_start 10.40.61.99
```

Figura 22: Comandos para ejecutar el RIS

Tras ejecutar estas líneas de comandos, y si todo está correctamente, nos aparecerá la ventana de inicio del RIS, mostrada en la Figura 23: Pantalla de inicio.

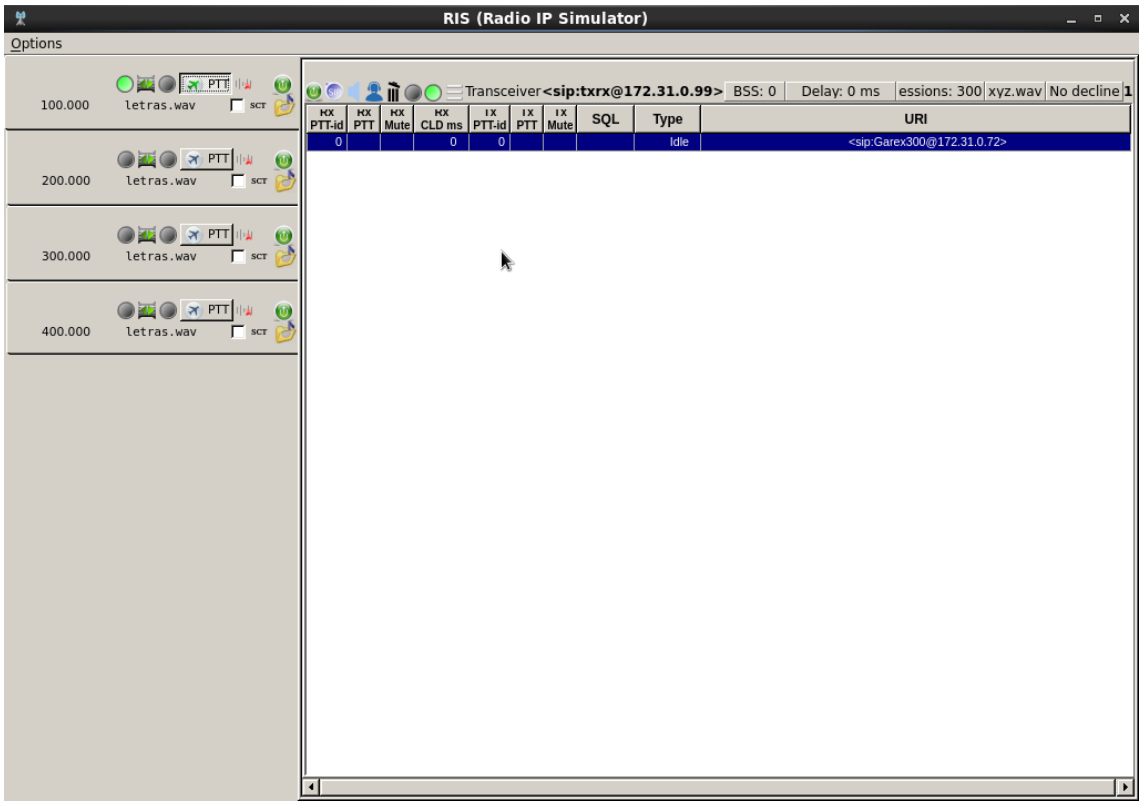

Figura 23: Pantalla de inicio

Centrándonos en la ventana en sí, podemos distinguir dos grandes partes. La primera ventana, donde se nos muestra información de los Equipos Radio (ER) simulados dentro de una misma Unión Radio (UR), las sesiones SIP que estos ER puedan tener establecidas y distintas funcionalidades que el usuario puede utilizar para modificar dicho ER. Una segunda ventana donde se nos muestra un listado de las Frecuencias de Control (FC) que el RIS ha simulado, indicándonos su frecuencia, su estado en tiempo real y ciertas funcionalidades explicitas para la FC.

4.2 Ventana de Equipos Radio

En este apartado vamos a centrarnos en todo lo mostrado en la ventana de ER.

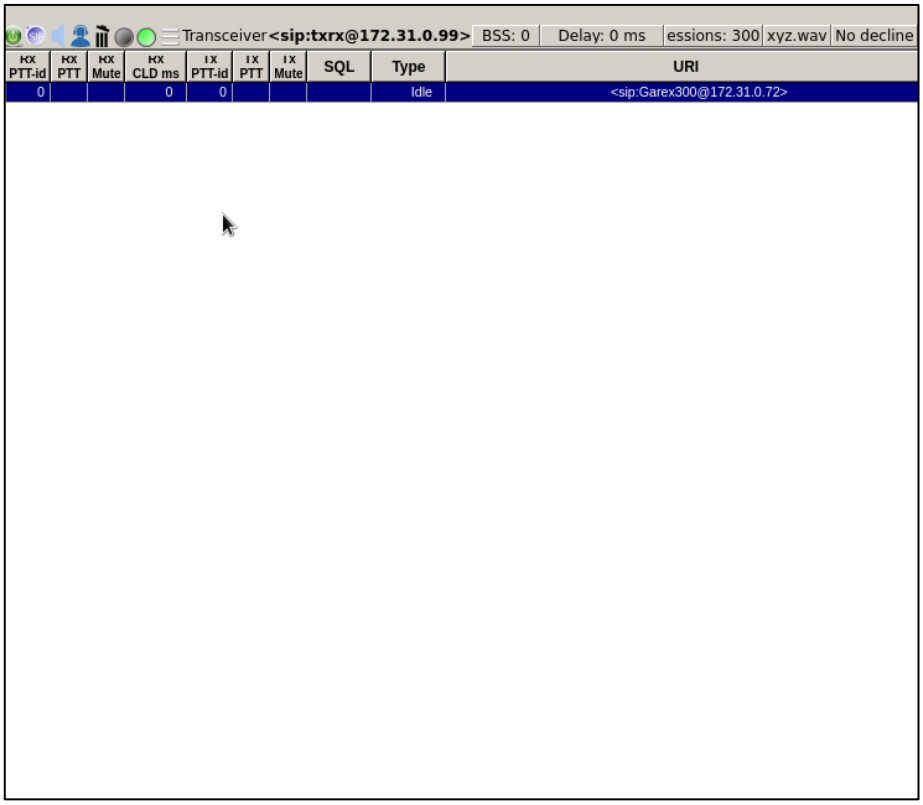

Figura 24: Ventana de ER

En la imagen de ejemplo, podemos ver una línea (marcada en azul) con toda la información actual de la sesión o sesiones SIP que tiene establecida este ER y la información también de los mensajes RTP que se mandan tras establecerse las sesiones SIP. Podemos ver en este mismo ejemplo, tratándose de un transceptor (TxRx), los campos de Rx PTT-id, Rx PTT, Rx *Mute*, Rx CLD ms, Tx PTT-id, Tx PTT, Tx *Mute*, SQL, *Type* y URI. Vamos a explicar a continuación estos campos:

- Rx PTT-id. Identificador que se le asigna al intercambio de mensajes RTP cuando hay una comunicación activa (se pulsa PTT)
- Rx PTT. Campo que se pone en verde si se recibe un PTT ajeno (recibiendo una comunicación)
- Rx *Mute*. Campo que se pone en verde si no se recibe audio por parte del transmisor de la comunicación.
- Rx CLD ms. Campo que indica el retardo que aplica la funcionalidad *Climax* (*CLimax Delay*).
- Tx PTT-id. Similar al campo Rx PTT-id, pero en este caso siendo el transmisor de la comunicación.
- Tx PTT. Campo que se pone en verde cuando se realiza una transmisión.
- Tx *Mute*. Similar a Rx Mute, pero en el caso de realizar nosotros la transmisión.
- SQL. Señalización de cuando hay *squelch* (SQ) en la comunicación.
- *Type*. Indica el tipo de la sesión SIP establecida. En el ejemplo, vemos que está en *Idle* (reposo).

- URI. Este campo nos indica la SIP-URI del equipo con el que tenemos establecida la sesión SIP.

Todos estos campos son de información de supervisión, lo que quiere decir que estos campos se actualizan en función de las modificaciones que reciba tanto el ER como las sesiones que tiene establecidas con otras máquinas.

Si nos fijamos en la parte superior, podemos ver una serie de iconos, los cuales, tienen las siguientes funciones:

Icono		Significado	Información
		*Power*	Sirve para encender o apagar el ER
		SQL	Sirve para habilitar o deshabilitar el SQ de una transmisión. Esto provoca que el receptor reciba el audio o no
		SIP/R2S	Sirve para habilitar o deshabilitar la mensajeria R2S que manda el ER para comunicar que sigue activo (*KeepAlive*)
		PTT *Confirmation*	Sirve para la confirmación de que el PTT ha llegado al receptor. En una transmisión radio, el receptor devuelve la señal portadora como confirmación de la correcta recepción del mensaje al transmisor. Deshabilitándolo conseguimos que el transmisor no sepa con seguridad si su transmisión ha sido exitosa o no
		Tx *Indicator*	Icono que se enciende o se apaga cuando hay una transmisión o no, respectivamente
		Rx *Indicator*	Icono que se enciende o se apaga cuando hay una recepción o no, respectivamente
		*Session BYE*	Sirve para rechazar una sesión SIP que tenga el ER establecida. Se puede elegir la respuesta que queramos enviar al servidor SIP indicando el motivo del *BYE* .

Figura 25: Iconos de la ventana de ER

El último botón, el botón de Menú, merece una mención distinguida debido a las distintas opciones que nos ofrece. Todas ellas, incluido los posibles valores que se puede elegir, son explicados a continuación en la siguiente tabla. Hay que tener en cuenta que en el ejemplo se trabaja con un transceptor (TxRx), por lo que en este menú estarán todas las opciones posibles, tanto para transmisores (Tx) como para receptores (Rx).

Icono	Opciones	Información	Valores
	BSS	Nos permite darle un número de identificación de BSS al receptor. El ID del BSS, al igual que el del PTT, sirve para diferenciar una retransmisión de otras. El BSS es una funcionalidad de los receptores para cuantificar la calidad de la retransmisión recibida, normalmente mediante señal/ruido (S/N) y dar esta información a la CWP para que esta pueda elegir la recepción de mejor calidad (*Best Sound Selected*).	Toma los valores de QID 1 hasta QID 15
	Rx *Delay*	Permite añadir un retardo ficticio al receptor. Pensado por si queremos utilizar la funcionalidad de *Climax*	Empezando en el valor 25 ms, da saltos de 50 ms hasta un máximo de 4975 ms
	Session Allowed	Permite elegir el número máximo de sesiones posibles para cada ER. Por defecto son 3000 sesiones.	Se puede configurar para que desde 1 sesión hasta 7.
	Climax (tj+tid)	Estos valores (tj+tid) hacen referencia a unos campos de tiempo que se envían en mensajería RTP para la funcionalidad *CLIMAX*. Estos campos concretamente hacen referencia al tiempo de propagación que hay entre la CWP y el receptor.	Empezando en el valor 5 ms, da saltos de 10 ms hasta 245 ms
	Audio to play	Aquí podemos elegir el audio que queramos transmitir cuando simulamos un PTT ajeno al sistema (por ejemplo, el recibido por un avión).	Archivos de audio `.wav` almacenados
	Decline	Similar a la función *BYE*, pero con la diferencia que, en este caso, elegimos la respuesta que se enviará ante la llegada de nuevos *INVITEs* para establecer una sesión SIP	Mismas opciones que la opción de "*Session BYE*"

Figura 26: Opciones del Menú de ER

Seguidamente de todos los iconos, siguiendo la Figura 25 se puede apreciar que nos indica que tipo de ER es. En este caso, vemos que pone Transciver ya que el ejemplo con el que estamos trabajando es un TxRx. Si fuese un Tx o un Rx, pondría Transmitter o Receiver respectivamente. Tras esto se nos indica en negrita la SIP-URI del ER que está simulando. Esta URI corresponde con las que hemos dado de alta en el TMCS al realizar la configuración. Y, para finalizar la barra superior, observamos que se nos indican los campos de BSS, *Delay*, *Sessions*, *Audio selected* y *Decline*. En estos campos podemos ver las características que hemos elegido en el menú y están actualmente seleccionadas para este ER.

4.3 Ventana de frecuencias de control

En este apartado vamos a explicar toda la información que se representa en la ventana de FC y las funcionalidades disponibles en la misma.

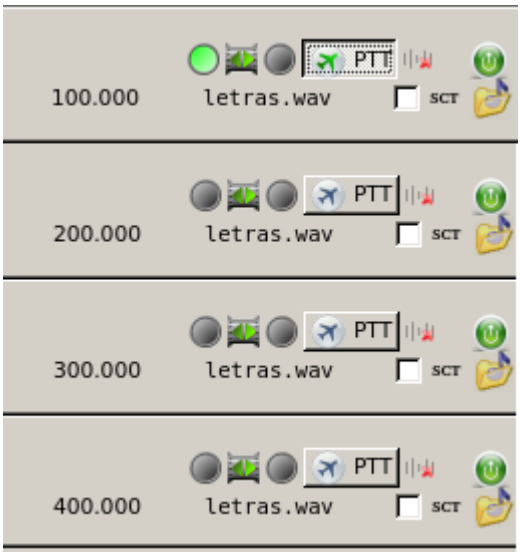

Figura 27: Ventana de FC

Observando la Figura 27 podemos contemplar varios iconos, así como la frecuencia que se ha configurado en el TMCS para que simule el RIS. En este caso, vemos que las frecuencias simuladas son 100.000, 200.000, 300.000 y 400.000.

Iconos		Significado	Información
		Rx Indicator	Icono que se enciende o se apaga cuando hay una recepción o no, respectivamente
		Break carrier return	Similar a la opción de PTT Confirmation pero, en este caso, el receptor no envía el la señal de retorno de portadora.
		Pilot PTT	Este botón nos permite simular que el Rx de la FC recibe una transmisión de un avión
		Audio detector	El SQ se puede configurar de dos maneras: hilo y audio. Configurado como hilo (primer icono), al detectar el PTT ajeno, el SQ se activará y se podrá escuchar la transmisión recibida. En cambio, configurado como audio (segundo icono), el SQ se activará solo al detectar los mensajes RTP con audio y desactivandose en los silencios.
		Power	Enciende o apaga todos los ER pertenecientes a dicha FC
		Audio files	Botón para elegir que audio queremos recibir cuando se active la opción de Pilot PTT. Tiene las mismas opciones que la opción "Audio to play" del menú

Figura 28: Iconos de la ventana de FC

5 Conclusiones y líneas futuras

5.1 Conclusiones

En esta memoria se ha diseñado, implementado y puesto en marcha una herramienta de simulación de radios IP, dotándolo de toda la funcionalidad necesaria para posibilitar la simulación de prácticamente cualquier escenario. Se ha diseñado un boceto inicial de la idea que se quería de la herramienta RIS. Con el conocimiento total del sistema Garex300 de Indra Sistemas S.A. y entendiendo las distintas funcionalidades que la tecnología RoIP nos ofrece, se ha comenzado con el desarrollo de la herramienta. Una vez conseguidos los códigos fuente para su correcta operativa, se ha creado una GUI lo más intuitiva posible para facilitar el uso de esta herramienta. Una vez conseguidos estos objetivos, se ha mostrado como exportar esta herramienta y explicar todas las funcionalidades implementadas en ella.

Gracias al desarrollo de esta herramienta, se pueden simular radios sobre IP para su uso durante la fase de pruebas de sistemas de comunicación por voz (SCV). Una radio IP real es un elemento costoso y, dependiendo del modelo, puede ocupar un gran espacio físico que hay que considerar a la hora de instalar el entorno de pruebas. El RIS es capaz de simular las radios IP correctamente y, además, cubre la totalidad de funcionalidades que la tecnología RoIP permite a fecha de hoy, como puede ser añadir retardos para simular un *Climax,* enviar distintas respuestas a mensajes enviados por el SCV contra el RIS, simulación de comunicaciones externas, etc.

En la actualidad, esta herramienta se utiliza diariamente en Indra Sistemas S.A. para probar exhaustivamente todos los escenarios posibles que el sistema Garex300 pueda tener con respecto a las radios IP. Gracias a la gran cantidad de ER que puede simular el RIS, se pueden realizar pruebas de carga del sistema Garex300, sobre todo del SERCOMAV, que de otra manera serían impensables o imposibles debido al elevado coste de las radios IP. Además, se está implementando la herramienta en el sistema COMETA, también desarrollado íntegramente por Indra Sistemas S.A., el cual se podría definir como el "hermano" del Garex300, pero cuyo cliente es una importante compañía española de gestión de tráfico aéreo.

La mayor pega que esta herramienta puede tener es su adaptación a otros sistemas que no sean el Garex300 o COMETA de Indra Sistemas S.A. Para poder realizar la adaptación es necesario conocer muy bien el funcionamiento interno de dicho SCV para hacer las modificaciones necesarias para su correcto funcionamiento.

En definitiva, el RIS es una herramienta muy completa hoy en día y que permite simular una gran cantidad de entornos de prueba, pero, con el paso de los años y con lo rápido que avanzan las tecnologías de telecomunicaciones, puede llegar a quedarse obsoleta. Es por ello por lo que es conveniente que sea "actualizada" según avance la tecnología RoIP.

5.2 Líneas futuras

El RIS es una herramienta muy completa por cómo es la tecnología RoIP hoy en día. Podríamos enfocar más este apartado hacia la parte visual de la herramienta, como puede ser mejorando el interfaz con opciones de búsqueda por frecuencia o por SIP-URI en caso de tener un gran listado de radios simuladas. También se puede mejorar el aspecto gráfico de la ventana, ya que es un poco rudimentario e intuitivo para un usuario sin experiencia en la tecnología RoIP.

En cuanto a funcionamiento, se podría añadir la opción de simular pruebas de carga. Esto quiere decir que, el RIS sea capaz de hacer simular transmisiones ajenas al SCV mediante radios simuladas específicas durante periodos de tiempo y que todo esto pueda ser editado por el usuario. Un ejemplo: las frecuencias 100.000 y 200.000 hagan simulación de PTT durante 20 segundos, reproduciendo un archivo de audio, y 40 segundos de silencio; mientras las frecuencias 300.000 y 400.000 hagan simulación de PTT durante 10 segundos, reproduciendo otro archivo de audio, y 50 segundos de silencio. Teniendo seleccionadas dichas FC en el SCV en modo transmisión y recepción, podemos ver cómo se comporta nuestro sistema durante largos periodos de tiempo.

Se podría aun dar más versatilidad en los escenarios del ejemplo anterior y que sea el propio RIS que, mediante un algoritmo de aleatoriedad, elija cuanto tiempo y qué radios van a simular los PTT y con qué archivos de audio.

Actualmente, las funcionalidades de RoIP están prácticamente en su totalidad implementadas en el RIS (se podría mejorar más la opción de BSS y/o Climax, que son las más complejas). La tecnología RoIP tiene aún mucho camino por recorrer por lo que seguramente su abanico de posibilidades aumente en unos años. Implementar las nuevas características que esta tecnología pueda ofrecernos hará que el RIS sea una herramienta más completa.

5. Conclusiones y líneas futuras

Bibliografía

[1] EUROCAE. *Interoperatibility Standars for VoIP ATM Components Part 1: Radio* [en línea]. Francia, 2009<http://startrinity.com/voip/Resources/ED-137_Interoperability_Standards_for_VoIP_ATM_Components_Part_1_Radio.pdf> [Consulta: 12 may. 2019]

[2] Indra Sistemas. *VoIP-Based voice communications control system* [en línea]. Madrid, España, 2013 <https://www.indracompany.com/sites/default/files/indra-voip-based_voice_communications_control_system_1.pdf> [Consulta: 15 jun. 2019]

[3] T. Berners-Lee. *Uniform Resourse Identifiers (URI)* [en linea]. R. Fielding, U.C. Irvine, L. Masinter. 1998 <https://www.ietf.org/rfc/rfc2396.txt> [Consulta: 18 jun 2019]

[4] J. Rosenberg. *SIP: Session Initianion Protocol* [en línea]. H. Schulzrinne, G. Camarillo. A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler. 2002 <https://tools.ietf.org/html/rfc3261> [Consulta: 18 jun 2019]

[5] H. Schulzrinne. *RTP: A Transport Protocol for Real-Time Applications* [en línea]. S. Casner, R. Frederick, V. Jacobson. 1996. <http://tools.ietf.org/html/rfc1889/> [Consulta: 19 jun 2019]

[6] Dieter Eier. *Eurocae WG-67 standars for voice-over-IP in ATM for advanced NEXTGEN conops* [en línea]. Wolfgang Kampichler. Herndon, VA, USA, 2010. <https://ieeexplore.ieee.org/document/5503299> [Consulta: 10 jul 2019]

[7] ITU. *G.711: Pulse code modulation (PCM) of voice frequencies* [en línea]. 1988. <https://www.itu.int/rec/T-REC-G.711-198811-I/en> [Consulta: 23 jul 2019]

[8] ITU. *G.729: Codificación de la voz a 8 kbit/s mediante predicción lineal con excitación por código algebraico de estructura conjugada* [en línea]. 2012. <https://www.itu.int/rec/T-REC-G.729-201206-I/es> [Consulta: 23 jul 2019]

[9] ITU. *G.728: Coding of speech at 16 kbit/s using low-delay code excited linear prediction* [en línea] 2012. <https://www.itu.int/rec/T-REC-G.728-201206-I/en> [Consulta: 23 jul 2019]

Apéndice A. Manual de usuario

Una vez hemos conocido todas las funciones y características, el usuario puede operar con la herramienta para simular los casos de prueba que quiera y comprobar cómo se comporta el SCV con el que está trabajando.

A continuación, vamos a recopilar los pasos a seguir para los diferentes escenarios de prueba que pueda encontrar el usuario.

A.1. Iniciar el RIS

Como hemos comentado en el apartado 4.1, la herramienta RIS tiene un script de arranque mediante terminal de Linux. Para ejecutar el RIS, introducimos las siguientes líneas de comando en el terminal.

```
cd / “ubicación deseada” /

./RIS_RIS_scr_unix_start 10.40.61.99
```

Figura 29: Comando ejecución

La IP que se introduce, que en el ejemplo es 10.40.61.99, debe ser la IP real de la máquina que nos encontremos y la misma que se han introducido en las URI-SIP en la configuración del TMCS y que se desee simular.

A.2. Seleccionar una FC

Por defecto, cuando arranquemos el RIS, aparecerá seleccionada la primera FC que esté en listado, que se corresponderá con la primera frecuencia creada en la configuración del TMCS.

Si lo que queremos es visualizar la información de los ER pertenecientes a otra FC, basta con hacer clic en el listado a dicha FC que se quiere usar.

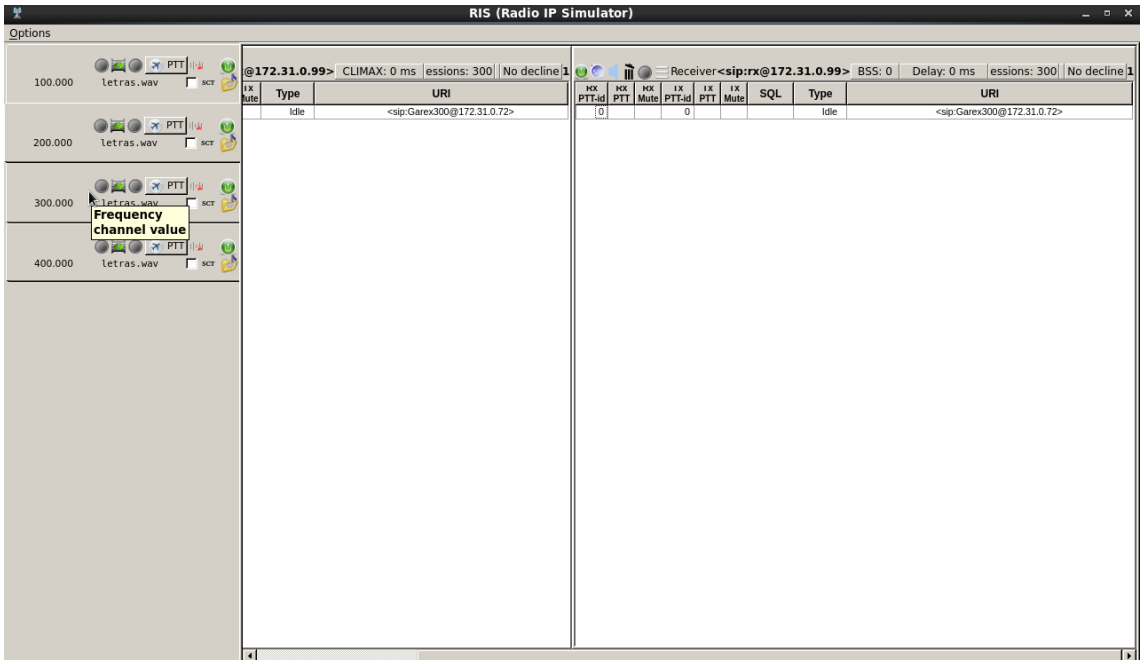

Figura 30: FC seleccionada

A.3. Apagar o encender un ER

Una vez seleccionada la FC, nos aparece la ventana con la información de los ER pertenecientes a dicha FC tal y como se muestra en la Figura 30.

Por defecto, a menos que se haya configurado de otra manera en el TMCS, todos los ER aparecerán encendidos. Para apagarlo, hacemos clic en botón "*Power*" y dicho ER se apagará.

Figura 31: Botón Power

Como se puede observar, el icono de "*Power*" actualiza su estado y se muestra la ventana del ER en un tono gris para notificar al usuario visualmente que no puede usar dicho ER.

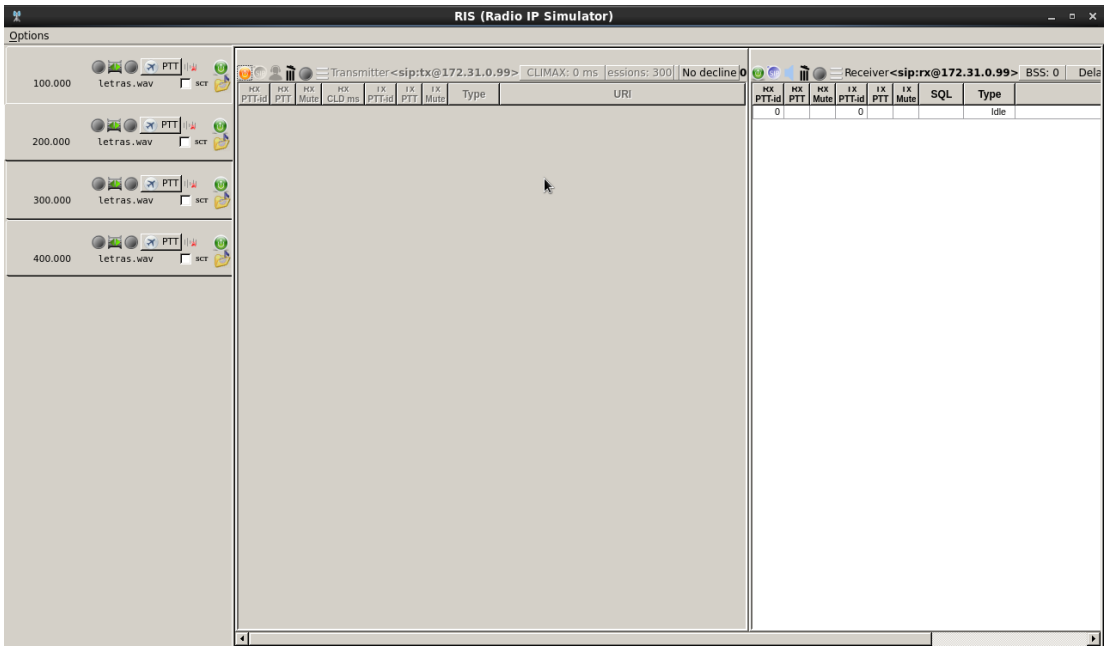

Figura 32: Apagado de ER

Para encender, solo tenemos que pulsar el mismo botón que para apagar. Cuando se encienda, el ER no tendrá ninguna sesión establecida aún, por lo que tendremos a que esperar dicho establecimiento SIP se cumpla para poder operar de nuevo con dicho ER.

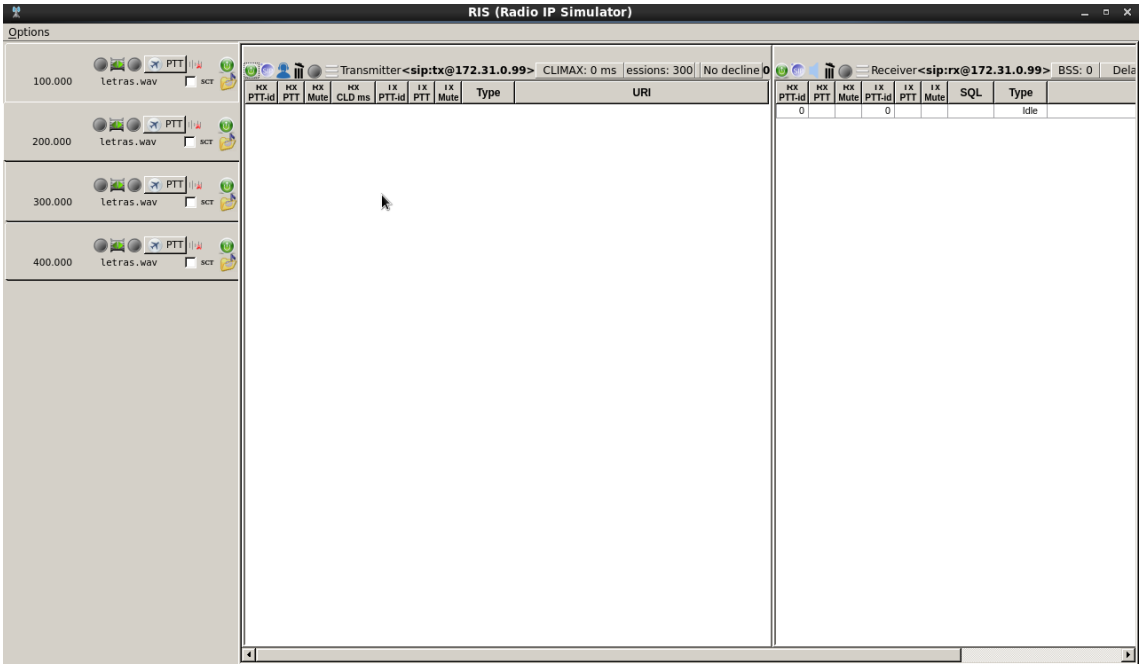

Figura 33: ER recién encendido

A.4. Habilitar o deshabilitar funcionalidades de un ER

Para habilitar o deshabilitar cualquiera de las funcionalidades explicadas en el apartado 4.2, basta con hacer clic al botón de la funcionalidad que se desee utilizar. Por ejemplo, si se desea usar la funcionalidad “SIP/R2S”, bastará con hacer clic en el botón.

Figura 34: Botón SIP/R2S

En cuanto lo hagamos, el icono del botón se actualizará a su nuevo estado (como se muestra dentro del círculo rojo de la Figura 35) y se aplicará la funcionalidad que, en este ejemplo, provocará la pérdida de todas las sesiones SIP establecidas.

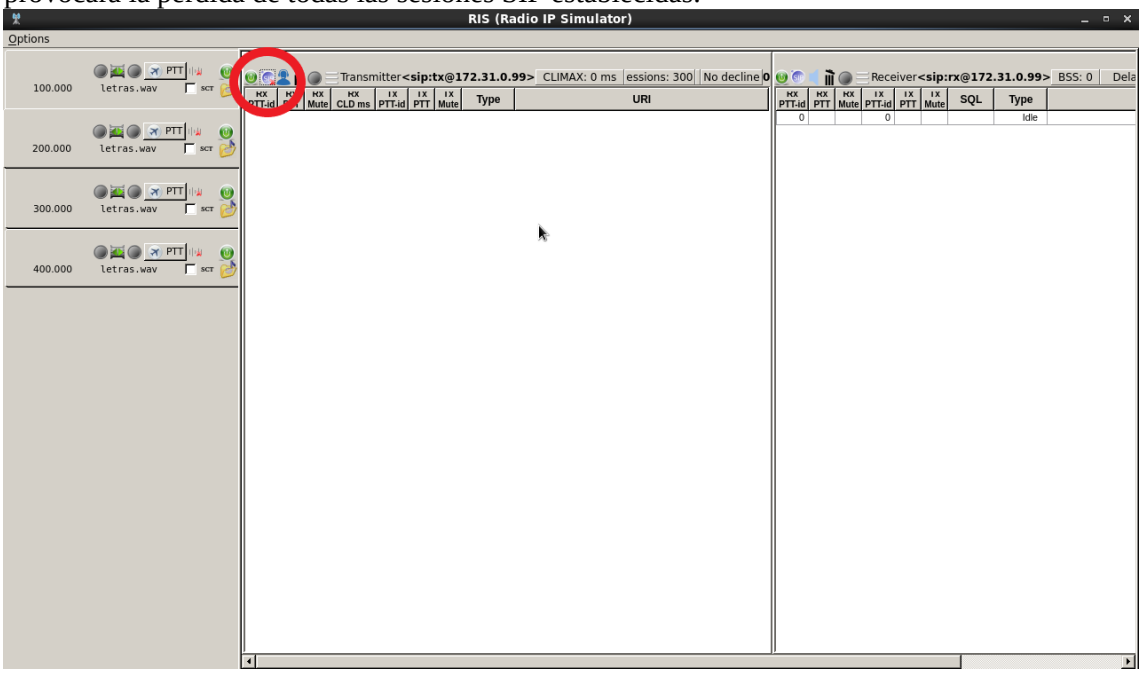

Figura 35: Funcionalidad SIP/R2S deshabilitada

Mención aparte merece la funcionalidad de “*BYE*”, que ofrece la posibilidad de elegir distintas opciones. Para ello primero debemos tener seleccionada la sesión SIP que deseemos eliminar. Basta con cliquear en la sesión SIP deseada seleccionarla.

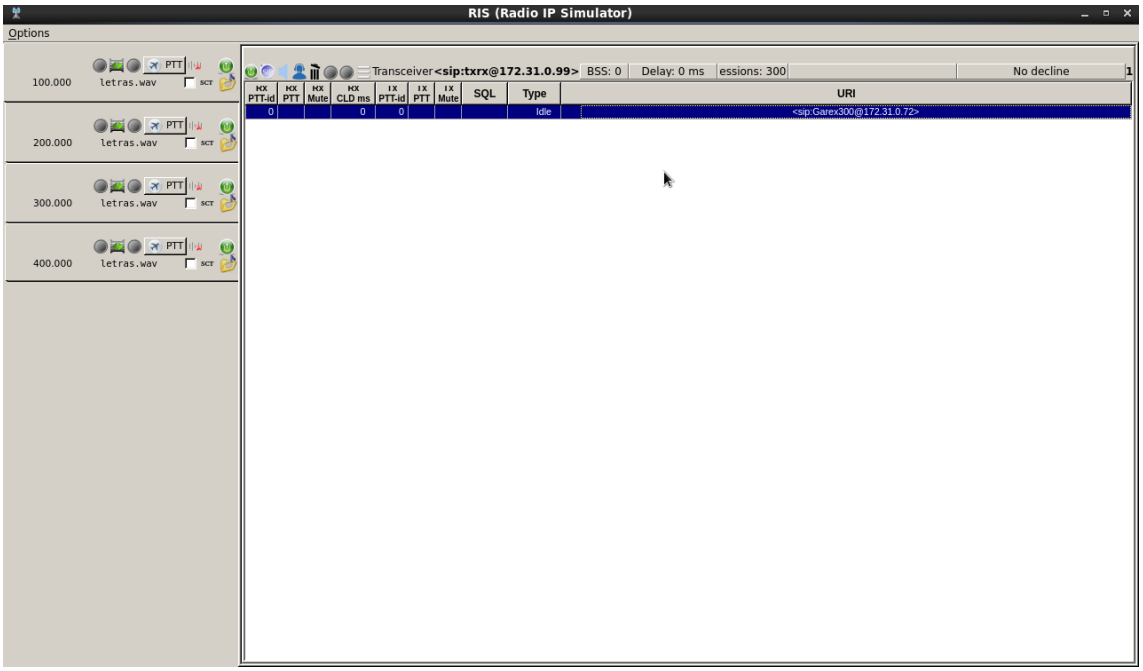

Figura 37: Selección de una sesión SIP en la ventana de ER

Una vez tenemos esta sesión seleccionada, procedemos a cliquear en el botón "*BYE*".

Figura 36: Botón BYE

Al seleccionarlo, nos aparecerá un listado desplegable con las opciones disponibles, tal y como se muestra a continuación.

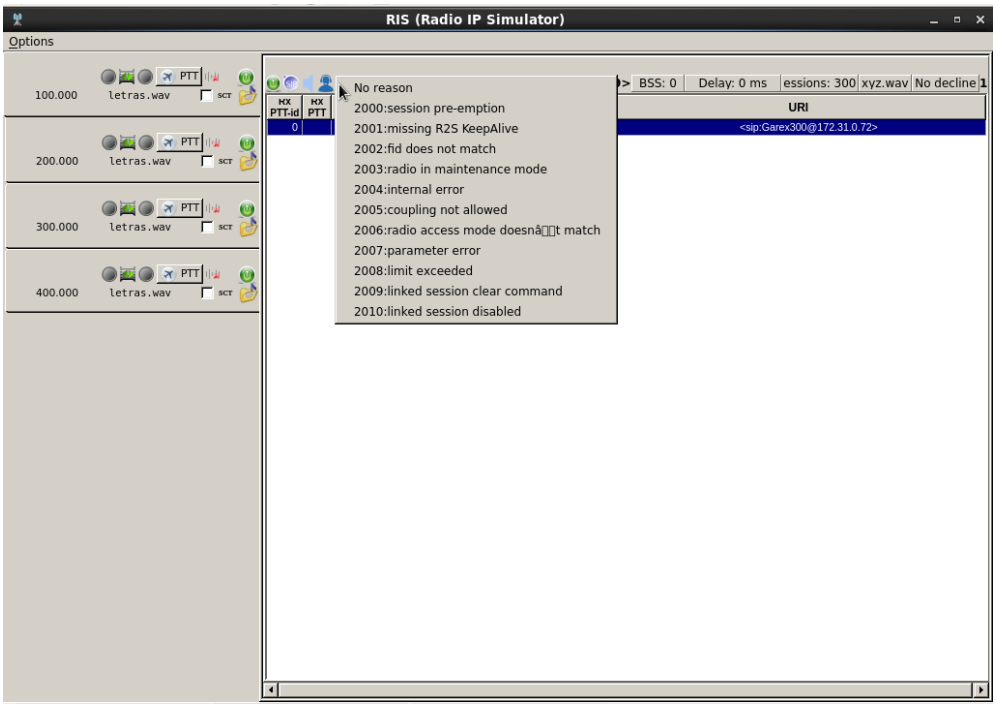

Figura 38: Listado de opciones de la funcionalidad BYE

Pasando el ratón sobre ellas, vemos que cambia de color la opción por la que tengamos el cursor encima. Para elegir una de estas opciones, solo hay que hacer clic en ella y se aplicará la funcionalidad. En este caso, la sesión SIP que hemos elegido será eliminada por la opción elegida en el listado desplegable.

A.5. Elegir una opción del menú de un ER

Como hemos visto en el apartado 4.2, en la ventana de cada ER existe un botón de “Menú”.

Figura 39: Botón Menú

Este botón, nos muestra varias opciones para editar las sesiones SIP establecidas con el ER. Las opciones dependen de la topología del ER, ya que hay opciones exclusivas para Tx como para Rx. Solo los TxRx poseen todas estas opciones, por lo que se ha elegido esa topología de ER para este manual.

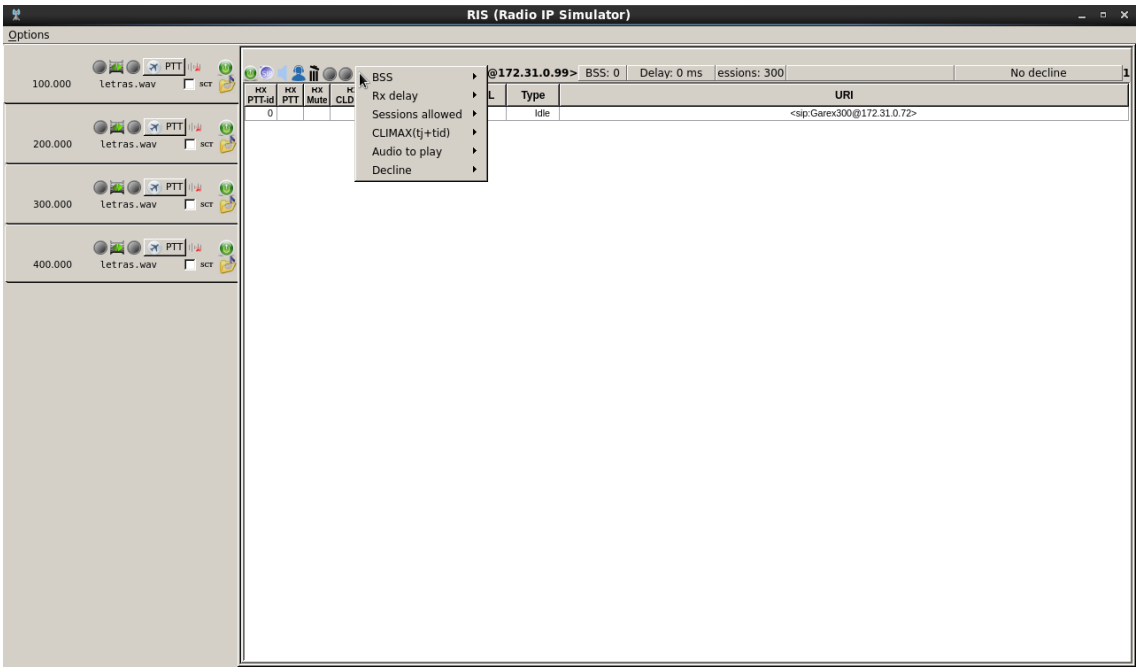

Figura 40: Listado de opciones de la funcionalidad Menú

Según pasemos el ratón por encima de este listado, nos aparecerá un segundo listado con los distintos valores que esa opción posee.

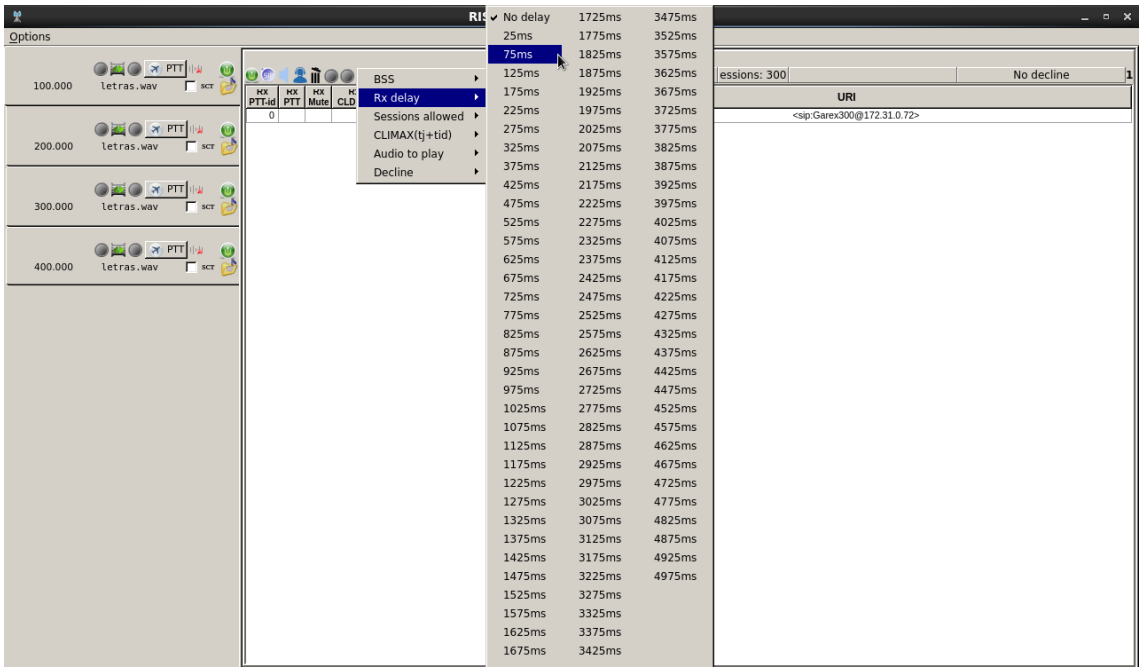

Figura 41: Valores posibles de la opción Rx delay del Menú

Basta con seleccionar con el ratón el valor que deseamos y se aplicara. Dicha modificación podrá verse en los elementos de supervisión que posee el ER. Para este caso de ejemplo, hemos elegido la opción "Rx *Delay*" del menú y le hemos dado el valor 75, como puede comprobarse en el círculo rojo de la Figura 42.

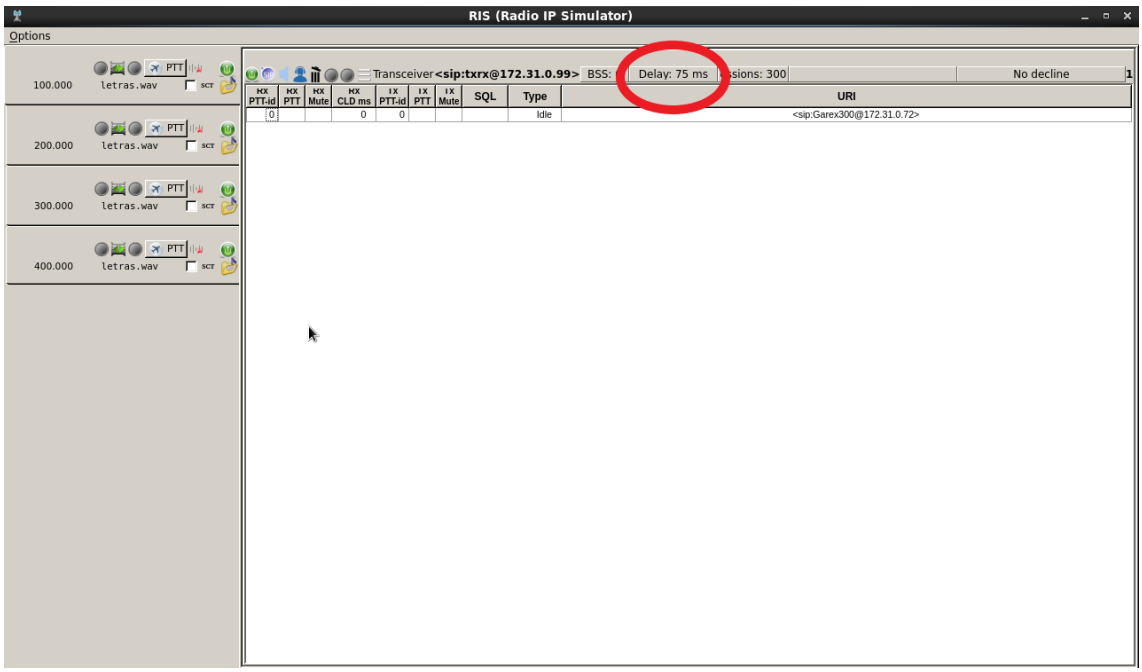

Figura 42: Supervisión al elegir la funcionalidad Rx Delay

A.6. Apagar o encender una FC

Teniendo una FC seleccionada del listado, procedemos a apagar una FC, que por defecto aparecerá encendida. Al igual que en el caso del apagado de un ER, solo hay que pulsar el botón "*Power*", visto en la Figura 31, para apagar todos los ER pertenecientes a dicha FC

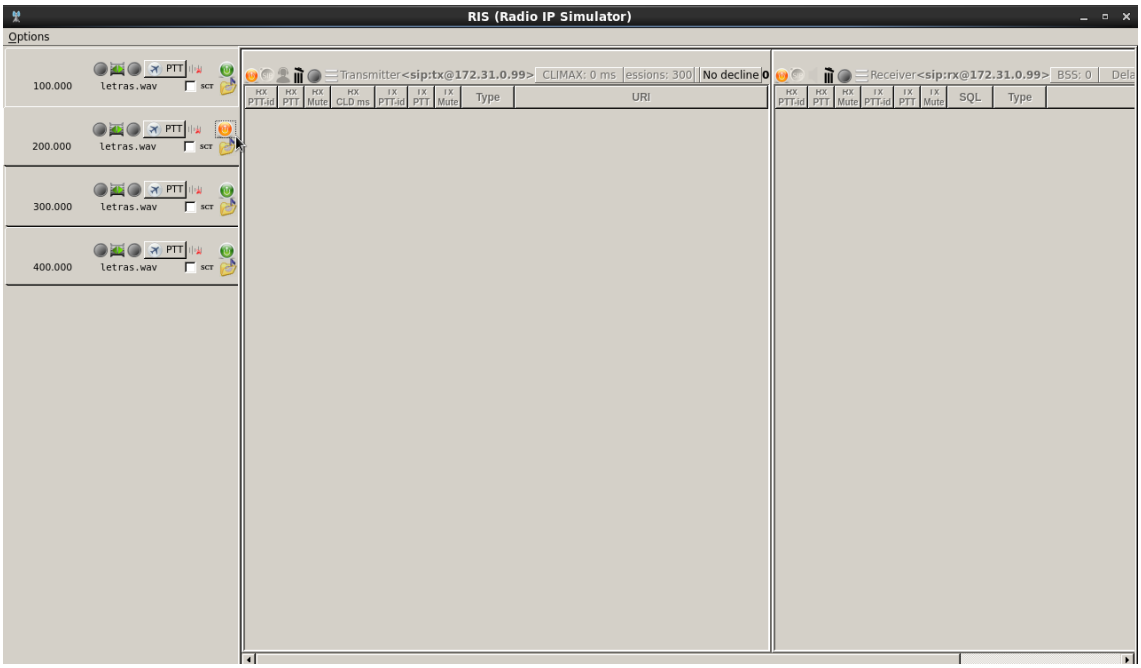

Figura 43: FC apagada

Como vemos, el icono de "*Power*" ha actualizado su estado a deshabilitado y todos los ER correspondientes se han apagado. Para encenderlo, volvemos a cliquear en el botón y los ER tornan a su estado original. Para que la FC esté otra vez operativa, hay que esperar a que se establezcan las sesiones SIP correspondientes de los ER.

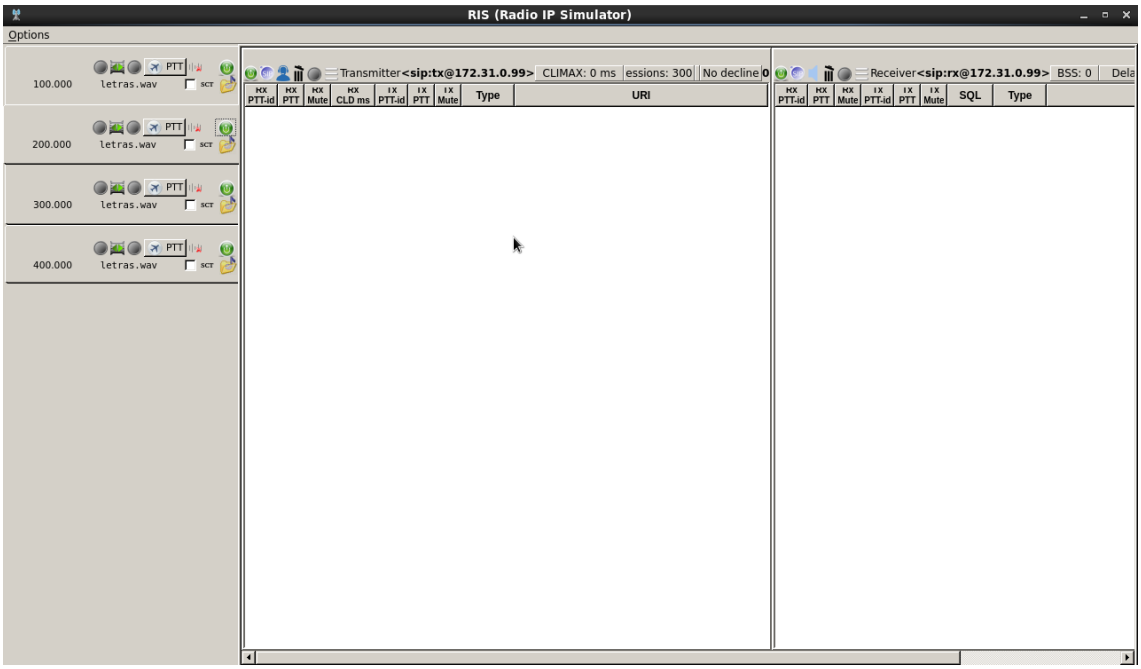

Figura 44: FC recién encendida

A.7. Habilitar o deshabilitar funcionalidades de una FC

Escenario similar al descrito en el apartado de funcionalidades de ER, ya que para activar o desactivar la funcionalidad que deseemos, solo hay que seleccionarla con el ratón. Por ejemplo, vamos a desactivar la funcionalidad de retorno de portadora nombrada "*Break Carrier return*".

Figura 45: Botón de Break Carrier return

Al hacer clic en ella, el icono actualiza su estado, como comprobamos en el interior del círculo rojo de la siguiente figura, y se deselecciona dicha operativa.

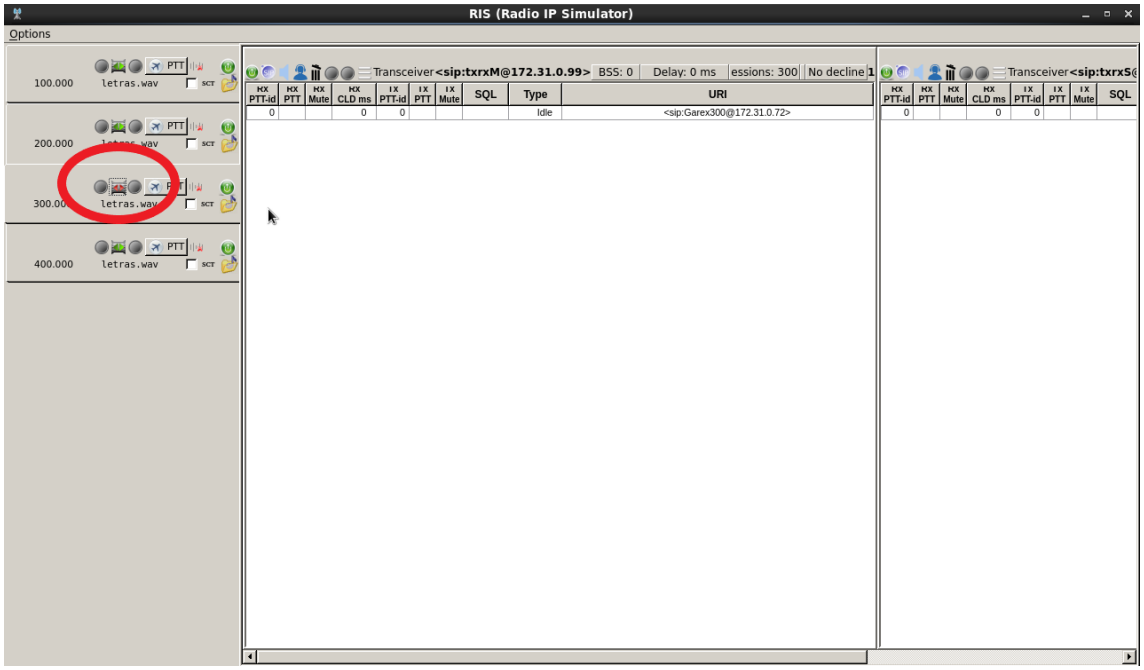

Figura 46: Funcionalidad Break Carrier return deshabilitada

Para volver al estado inicial, repetimos el proceso y cliqueamos el botón de "*Break Carrier return*".

A.8. Elegir o añadir un archivo de audio a la FC

La funcionalidad para elegir el archivo de audio, nombrada "*Audio files*" tiene la especialidad que nos ofrece elegir entre distintos archivos.

Figura 47: Botón de archivos de audio

Haciendo clic en el icono, nos mostrará un listado desplegable con todas las opciones disponibles.

Al pasar el ratón por encima del listado, se irá marcando con distinto color la opción sobre la que se sitúe el cursor. El usuario elige el archivo de audio que quiera asignar a dicha FC con solo seleccionar la opción que desee. Tras esto, el listado desaparecerá y en la tecla de la FC se podrá ver el archivo de audio que se ha elegido.

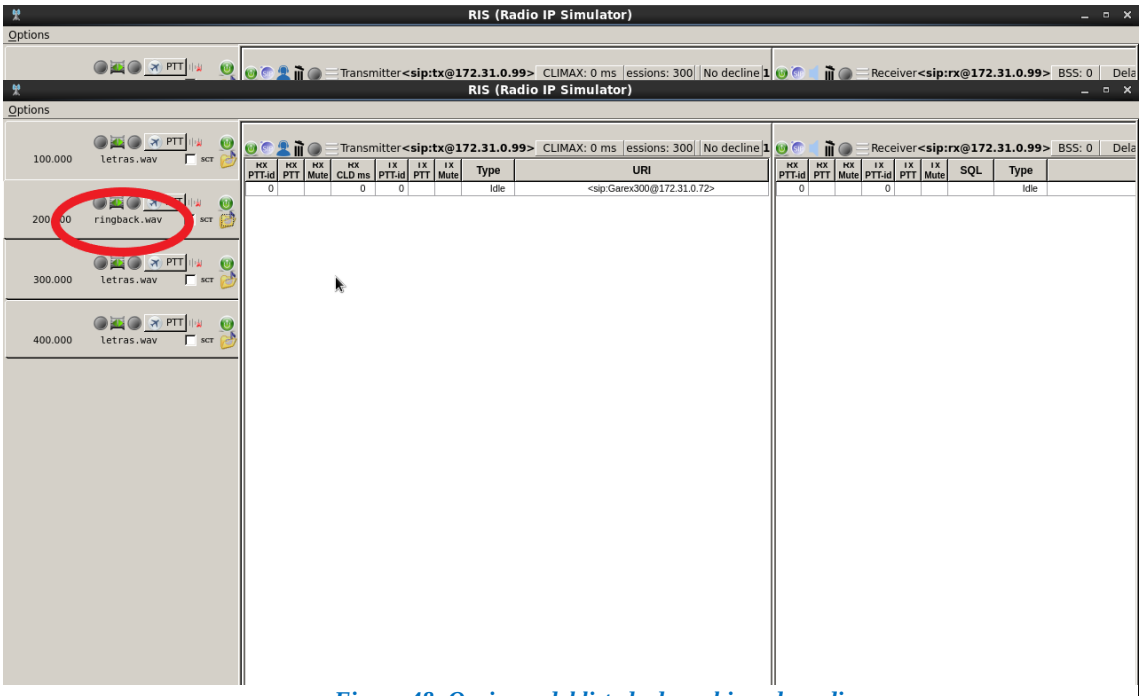

Figura 48: Opciones del listado de archivos de audio

Si el usuario desea añadir archivos de audio a la herramienta RIS, debe añadirlo a la carpeta "snd" que encontrará en el destino donde haya sido descomprimido el RIS.

El archivo de audio que se quiera añadir debe ser del formato "`.wav`" para que el RIS lo reconozca y pueda ser utilizado por el usuario.

```
cd /"ubicación donde se encuentre el archivo de audio"/

cp "archivoaudio.wav" / "ubicación del RIS" /snd/
```

Figura 50: Añadir un audio

A.9. Usar la funcionalidad PTT Pilot

La funcionalidad de "PTT *Pilot*" tiene la misma operativa que el método explicado en el apartado A.7.

Figura 51: Botón PTT Pilot

Esta operativa puede ser la más útil para el usuario, ya que le permite simular un PTT ajeno al sistema. Al hacer clic en la funcionalidad, el archivo de audio que esté seleccionado en la FC será el que se reproduzca en el SCV. Si no se ha seleccionado ninguno, por defecto se tiene seleccionado el archivo "`letras.wav`". Si en cambio, se ha seleccionado un archivo de audio en el ER Rx, tiene prioridad esta selección de audio que la correspondiente elección de audio hecha en la FC.

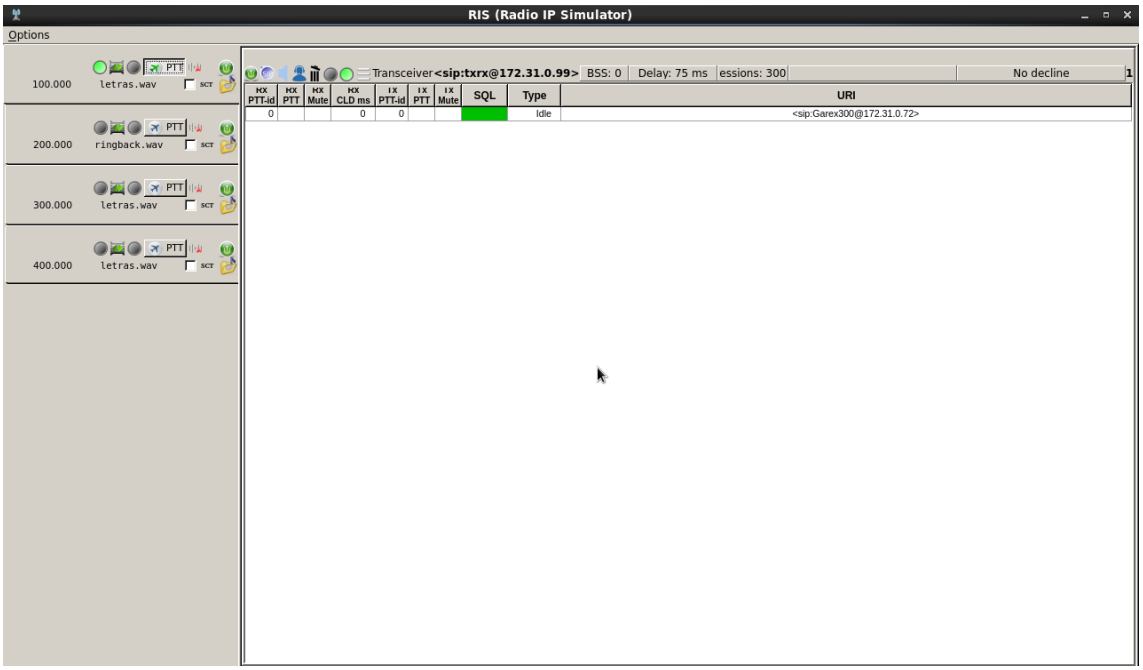

Figura 52: Funcionalidad PTT Pilot activa

Al activar la funcionalidad, el icono de “PTT *Pilot*” pasa a estado activo y se encienden los leds que indican la recepción en el ER Rx o TxRx.

Si volvemos a pulsar el botón, se deshabilita la operativa y todo retorna al estado inicial.

A.10. Aumentar o disminuir la ventana del RIS

Si el usuario desea aumentar la ventana del RIS para poder ver más información de la ventana de ER o, en su defecto, desea minimizar la ventana para que el RIS siga ejecutándose y tener el escritorio libre, tiene la disponibilidad de hacerlo con los iconos que encontrará en la esquina superior derecha de la ventana del RIS.

Figura 53: Botones de gestión de ventana

El botón que se encuentra a la izquierda minimizará la ventana y la llevará a la barra de tareas. En cambio, si pulsa el botón central, modificará el tamaño de la ventana. Con el primer clic, ampliará el tamaño de la ventana.

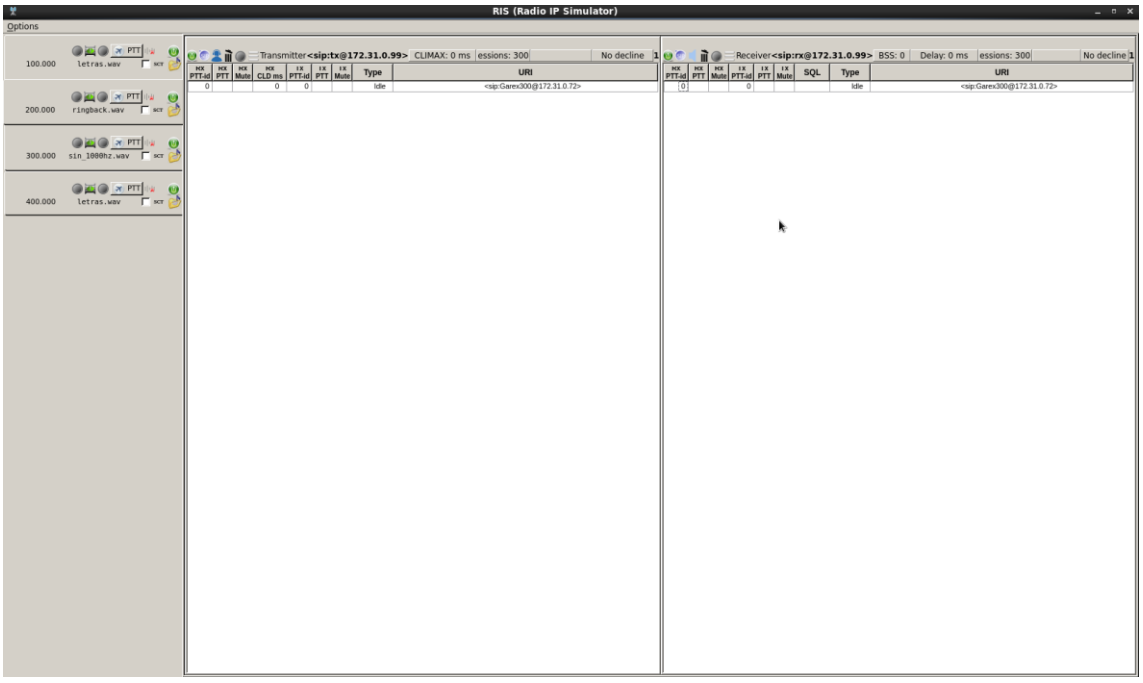

Figura 54: Ventana ampliada

Al volver a cliquear en el mismo icono, la ventana volverá a su tamaño original.

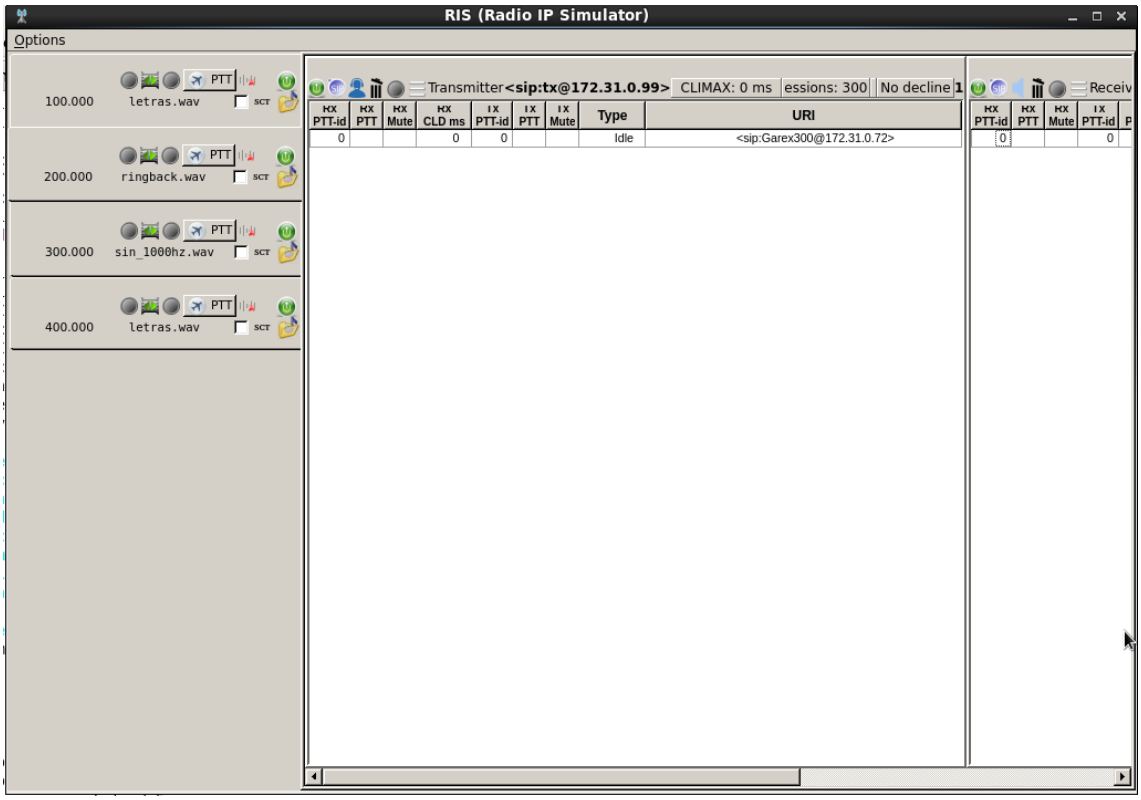

Figura 55: Ventana minimizada

Si el usuario desea regular el tamaño de la ventana, puede hacerlo situando el ratón a un borde de la ventana. Al hacerlo, el logo del cursor cambiará y bastará con cliquear y arrastrar para agrandar o disminuir la ventana al tamaño deseado.

A.11. Cerrar el RIS

Para cerrar la ventana del RIS, y, en consecuencia, la simulación de las radios IP, el usuario hace clic en el aspa mostrada en la Figura 53.

Una vez hecho, la ventana desaparecerá y se matará el proceso de la simulación llevada a cabo por el RIS. Para volver a arrancar la herramienta de nuevo, hay que repetir la operativa explicada en el apartado A.1.

Universidad de Alcalá

Escuela Politécnica Superior

ESCUELA POLITECNICA SUPERIOR